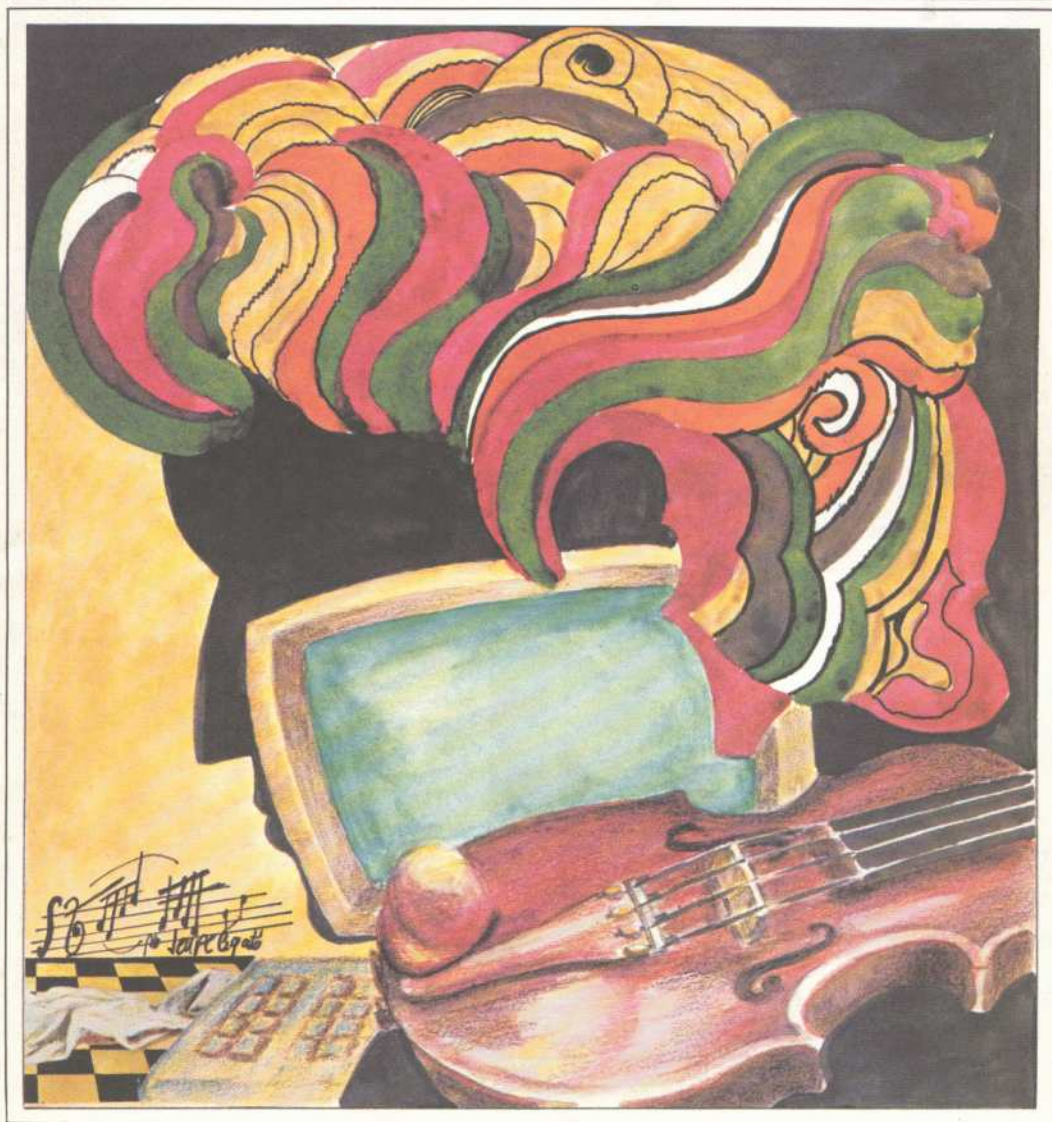


mikro

számítógép

magazin

Ára: 30 Ft

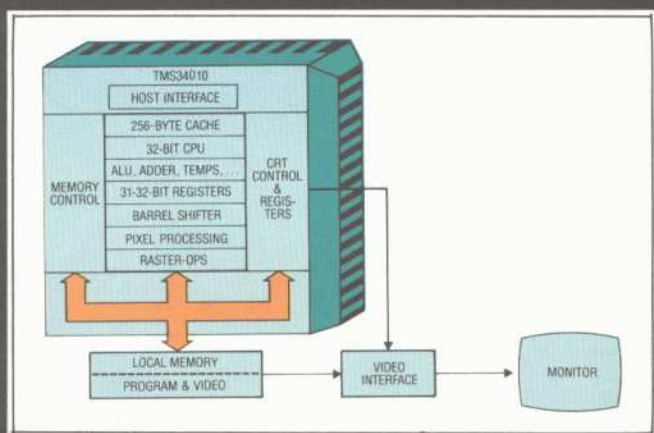


TARTSUK A TEMPÓT!

1989/4



VGA Plus 16 kártya



*TMS34010
processzoros rendszer tömbvázlata*

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

A szerkesztőbizottság vezetője:
Kovács Győző

A szerkesztőség munkatársai:
Bakos Tamás
(programozástechnika)
Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Nagy Imre
(tanuljunk együtt)

Petróczy Judit
(könyvek)

Pinke György
(Enterprise)

Soltészné Vizi Zsuzsa
(tervezőszerkesztő)

Simonyi Endre

Szebzenski Sándor
Szulyovszky Csaba
Tamásné Lakó Erika
Terebessy Ákosné
Varga János

Cimképünk:
Velekey József Lajos
munkája

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433

Kiadja:
MTESZ Neumann János
Számítógéptudományi Társaság
1054 Budapest, Báthori u. 16.

Levélcím:
1368 Budapest 5. Pf. 240

Telefon: 329-349

Felelős kiadó:
Tóth Istvánné ügyvezető
főtájkárhelyettes

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő
hivataloknál
és a Posta Hírlap-előfizetési
és Lapellátási Irodáján
(1900 Budapest XIII.,
Lahel u. 10/A)
vagy átutalással a 215-96 162
pénzforgalmi jelzőszámra.

Megjelenik havonta.
Egy szám ára 30,- Ft
Előfizetési díj:
egy évre 360,- Ft
fél évre 180,- Ft
Külföldön terjeszti
a Kultúra,
1389 Budapest, Pf. 149
és a Magyar Média
1932 Budapest, Pf. 279
88-1552



Szikra Lapnyomda
Budapest (89-0316)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

TARTALOM

2	Levél a kórházból II.
10	Feladatok — megoldások
27	Rendszerfejlesztési eszközök
31	Merre tart a világ?
34	A felvirágzott háttértároló
40	Olvastunk ...
43	Programtermék
46	Adok — veszek — cserélek

TANULJUK EGYÜTT!

3	A Pascal rejtelmei
6	Tette kész mindenes
7	Tempótartás kopogással
8	„KEYBOARD”. Hangos billentyűzet
8	„SEARCH.S”. Hangos keresőprogram
9	Az írógéptől a számítógépig
10	Kettőt egy csapásra

CSIPEGETŐ

13	The last ninja II.
14	Belelélelőtől a tálalásig
15	Örökélet
15	Fele(más) nyomtatás
16	„Átütetés” gyerekeknek is
16	TOP-lista

PROGRAMOZÁSTECHNIKA

17	A számítógépek motorja III.
18	Legyen emberközelí!
19	Programozási fogások és melléfogások
20	Legyen-e sorellenőző kód?
21	Rekurzió átalakítása iterációvá

ENTERPRISE

23	A BASIC programok tárolása a memóriában
24	Függvényeink függvényében
26	Mi a manó?

PROGRAMOK

36	Konverziós program IBM PC-re
----	------------------------------

KLUB

38	A The Newsroom magyar változata II.
39	Adom a magyarázatot!

SAKK

44	A figuratípusok jutalompontjai
----	--------------------------------

KÖNYVEK — HÍREK — ÉRDEKESSÉGEK

AZ OLVASÓ ÍRJA

μ mikro számítógép magazin



ARTIST: A. TEMPÓI

3

13

17

23

36

38

44

46

48

„Minden ember életében van olyan időszak, amikor a balszerencéből valami jó fakad. Amikor a dolgok olyan baljós fordulatot vesznek, hogy az ember nem tehet mást: üstökön ragadja a sorsot és jól megrázza.”
(Lee Jacocca: Egy menedzser élete)

Úgy látszik, hogy a kórász levelű műfaj most már évente előjön az életemben, pedig nagyon nem szeretem a publicisztikának ezt a formáját. Különösen akkor nem, ha mindez a saját ostobaságom következménye: egy rizikómentes előzés — gondoltam én —, frontális találkozás egy teherautóval — határozott a sors. Szerencsére a mellettem ülő fiatal kollégámmal együtt megúsztuk. Hál'istennek ő kissebb sérüléssel, én egy kicsit bonyolultabb módon, de azért remélem, hogy elfogadható idő alatt ismét mozgékony leszek.

Eddig persze mindez magánügy, de közben a Mikroszámítógép Magazin

Nagyon nem szeretném, ha ez az írás nyafogásra sikerülne, de a tudományos egyesületek jövőjét, anyagi lehetőségeit egyre nehezebbnek látom. Sőt, azt is megkockáztatom, hogy a szakvállalatok anyagi és erkölcsi segítségére nélkül bizonyos feladatokat egyszerűen lehetetlen lesz teljesíteni.

januárjában lett 25 esztendő a Neumann János Számítógéptudományi Társaság is.

Ezért javasoltam — ha már az évfordulók ilyen jól összejötték, és ráadásul sikerült megőriznünk egy csomó öreg számítógépet is —, tiszteljük meg a szakmát egy állandó kiállítás létrehozásával. A Szabolcs-Szatmár megyei vezetők ajánlkoztak társul a vállalkozáshoz, és vállalták egy ilyen kiállítás költségeinek legnagyobb részét is. De azért legyünk őszinték, ez az ügy nem csak Szabolcs-Szatmár megye ügye. Amikor a körlevelet a megyei szervezeteknek elküldtem, a megyéknek egy kis része sajnálkozott, hogy nem tudja a feltételeket biztosítani, a nagyobbik része pedig még csak nem is válaszolt. E tények tudatában a szabolsiak felajánlását különösen meg kell becsülnünk.

A kiállítás költségeit nyilván nem lehet csak a megyére hárítani. Amit még össze kellene szednünk, az hozzávetőlegesen 3 millió forint, hogy a gépeket a helyszínre szállíthassuk, a tablókat elkészítsük, egyezőval a kiállítást méltóvá tegyük az évfordulóhoz. Ezt a 3 millió forintot ismét csak az NJSZT kétszáz feletti jogi tagközösségétől kértem, örök optimistaként azt gondolva, hogy egy ilyen akció nem csak a sziveket, de a bukszákat is megnyitja. A körlevélre összesen öt cég válaszolt. Az MTA és a Comporgan jelentős összeget, a másik három néhány ezer forintot ajánlott, illetve biztató szavakat küldött támogatóként.

En persze érthetetlennek tartom a többi kétszáz-egynéhány cég magatartását. Az ugyanis elképzelhetetlen számomra, hogy néhány tízezer forint annyira a tőnk szélére taszítana egy-két országos nagyvállalatot, hogy még csak nem is válaszolnak. Vagy — ez a másik lehetőség — annyira érdektelen ma a vállalati szféra az ilyen társadalmi ügyek iránt, hogy nem is szabad támogatásért hozzájuk fordulni.

Szóval fekszem az ágyon itt Dunaújvárosban, és élvezem a korábbi kormányzat döntéseinek eredményét a hazai telefonhívással kapcsolatban. A kórászból ugyanis alig negyven perc alatt tudom például a Vasművet megkapni, Budapestet viszont kora reggel vagy nagyon késő este, akkor is csak igen nehezen. Tehát úgy el vagyok vágyva a világtól, mintha egy korallzátonyon élnék, közben az idő megy, és gyakorlatilag nem történik semmi.

Most azon gondolkodom, hogy nem lettem-e pesszimista egy autóbaleset következtében. Vagy ma ez a realitás?

Kovács Győző

Levél a kórászból II.

várja a szerkesztőségi cikket, és most — sajnos — van elég időm töprengeni a számítástechnika ügyes-bajos dolgain, a kényszerű pihenőm alatt.

A gondolkodásomat segítette, de a hangulatomat nem nagyon emelte, hogy néhány napig mozdulatlanul kellett feködnöm, mert a lábamra gipszkolonc került. A második napon pedig már úgy éreztem, hogy a hátam kilyukad a sok fekvéstől, és akkor még a rendszeresen visszatérő szurkálások örömeit nem is említettem.

Az sem tesz túlságosan vidámmá, hogy éppen ma kezdődik az NJSZT Országos Elnökség ülése, amelyet másfél-száz kilométerről próbálok követni, ami persze lehetetlen, de legalább gondolatban ott vagyok, és Balatonföldváron kinomban egy nyílt levelet írtam az elnökség tagjainak.

Azt hiszem ugyanis, hogy nagyon fontos kérdésekről van szó. Különösen ma, amikor egyre inkább anyagiasodó, profitcentrikus világban élünk, és végül is azt kell az NJSZT vezetőségének elhatároznia, hogy a társaságot egy nyereségorientált vállalkozássá alakítja át, ahol a prioritásokat a jövedelmezőség határozza meg, vagy pedig megpróbálja a társadalmi célokat követni, és támogatások útján a megfelelő anyagiakat megszerezni.

En ahhoz a kisebbséghez tartozom, amely az utóbbi elképzelésekkel ért egyet, annak ellenére, hogy tudom, az állam a támogatásokat az élet szinte valamennyi területén mérsékli vagy teljes egészében megszünteti.

Voltaképpen arról is szó lesz, hogy ma hogyan tudják a tudományos egyesületek, például a Neumann János Számítógéptudományi Társaság is, a működésükhöz szükséges anyagiakat előteremteni akkor, amikor a bevételi forrásai, azaz az egyéni tagdíjak és a rendezvények alaposan megadóztatott bevételei egyre szűkülnek. Hiszen a pénzügyi kormányzat a tudományos egyesületeket sem kezeli másként, mint a termelővállalatokat.

A közelmúltban két kísérletem is volt — az egyik az írás pillanatában még tart —, hogy „magasztos” egyesületi céljaink teljesítéséhez társadalmi támogatást szerezzek. Az egyik éppen a Mikroszámítógép Magazinak a jelenlegi formában való továbbléte, a másik pedig a nyíregyházi állandó számítástechnika-történeti kiállítás létrehozása volt.

Azt minden olvasónk tudja, hogy a Magazin folyamatosan anyagi nehézségekkel küzd: tudniillik a jelenlegi 30 forintos példányonkénti ár a nyomda-, papír- és szerkesztőségi költségeket nem fedezi. Valószínűleg nem halogathatjuk tovább a lap árának a mintegy 50 százalékos emelését, mert az egyre fokozódó mértékű inflációt sajnos követnünk kell, de még így sem tudjuk a lapeladásból a költségeinket teljes mértékben fedezni. A Mikroszámítógép Magazin az amatőrök, az ifjúság lapja, még akkor is, ha az előfizetőink egy igen kis hányada az érdeklődő felnőttek közül kerül ki. Végeredményben tehát — nekem ez a véleményem — a lap árát nem az előállítási költségre, hanem az olvasók jövedelmi viszonyaihoz kell igazítani, és a veszteségeket más forrásból kell fedeznünk. A lap pénzügyi egyensúlyát könnyen helyreállíthatnánk például, ha a Neumann társaság jogi vállalatokká vállalnák, hogy évente egyszer-kétszer hirdetés nélkül. Ez az árbevétel a veszteségnek teljes mértékben fedezné.

Ugyanígy — a társaság költségvetésében — egyelőre nincs fedezet a nyíregyházi hazai számítástechnika-történeti kiállítás létrehozására sem. Ha valaki nem tudná: 1959. január 29-én jelent meg az újságokban, hogy elkészült az első magyar elektronikus számítógép, vagyis 1989. január 28-án volt 30 éves a magyar számítástechnika. Ráadásul Tóth Endre, az Országos Műszaki Múzeum főműzeológusa a korabeli dokumentumokból kiderítette, hogy 1964. január 27-én alakult meg a MTE SZ keretében az IKOSZ, az NJSZT elődje, az Információfeldolgozási, Kibernetikai és Operációkutatási Szakosztály, tehát ez év



A PASCAL REJTELMEI

6. Negyedik programunk

Talán nem merészség azt feltételezni, hogy sorozatunk előző részeinek elsajátításával olvasóink már kellő nyelvismerettel vértézhetnék fel magukat. Eppen ezért a továbbiakban komolyabb feladatokra vállalkozhatunk, ugyanakkor mélyebbre hatolhatunk a Pascal titkosabb régióiba is.

Ezúttal tíz integer típusú adat (szám) növekvő sorrendbe rendezése és a képernyőre kivitele lesz a feladatunk. A rendezendő adatokat egy tömbváltozóban helyezzük el. Egy tömbváltozó deklarációja a következő formájú:

```
var tomb:array [min..max] of  
típus
```

Ha a tömb elemei — típusai — standard típusok, külön típusdeklarációra nincs szükség. Az **array** kulcsszót követő szögletes zárójelben a tömb első és utolsó elemének indexét kell megadni. Az index értéke csak **integer** szám lehet.

Részletesen foglalkozunk a globális és lokális változókkal és a változók érvényességi (hatás-) körével. Először fogjuk alkalmazni a feltételes vezérlésátadás legegyszerűbb eszközét, az **if..then** utasítást.

6.1 Globális és lokális változók

Eddigi programjainkban a felhasznált változókat a program elején, a programfejet közvetlenül követő deklarációs részben definiáltuk. E változók a program bármely részében — az eljárásokban is — alkalmazhatók voltak, értékükhöz hozzáférhettünk. Az ilyen változók érvényességi köre tehát az egész

program, elnevezésük globális változó.

A Pascal lehetőséget ad arra is, hogy változókat deklaráljunk a programhierarchiában tetszőleges helyen lévő eljárásokban is. E változók hatásköre csak az adott vagy az abból hívott — a hierarchiában alacsonyabb szinten lévő — eljárás. Ezek a változók a program magasabb szintjein vagy a programfa más ágain lévő helyekről, például eljárásokból nem érhetők el; ha rájuk hivatkozunk, akkor **unknown identifier** (ismeretlen azonosító) hibajelzést kapunk. Az ilyen módon deklarált változók neve lokális változó.

A globális és lokális változó meghatározást természetesen rugalmasan és értelemszerűen kell alkalmazni: például egy eljárásban deklarált változó az adott eljárásból hívott minden eljárás számára globálisnak tekintendő.

Gyakran előfordul, hogy egy változó érvényességi körén belül, történetesen egy hívott eljárásban, a változót más tartalommal ruházzuk fel, egy új deklarációval újradefiniáljuk. Ilyenkor az új deklaráció hatáskörén belül a változó eredeti értékét elveszti, majd abból kilépve, azt visszanyeri.

Az elmondottakat illusztrálja a 15. ábrán látható program, a futás eredményét pedig a 16. ábra.

Az **i** az **újradeklarál** program globális változója. Az **ujra** eljárásban mint lokális változót újradefiniáltuk és **i**-nek 1 értéket adtunk.

A főprogram kezdetén az **i** értéke 2 lesz. Ezt kivisszük a képernyőre, majd az **ujra** eljárást hívjuk. A kivetelkor a képernyőn most 1 jelenik meg. A program következő **writeln**-je már az újradeklarálás

hatáskörén kívül van, tehát az **i** visszakapja eredeti értékét (2), és kiírásra most ez kerül.

A figyelmes olvasó ezek után joggal veheti fel: a gyakorlati problémák megoldása során csak ritkán alkalmazunk „elszigetelt” lokális változókkal dolgozó eljárásokat, majd minden esetben szükség van a hívó és a hívott programrészek közötti adatcserére is! A Pascal erre is igen egyszerű lehetőséget nyújt; ezzel a következő részben, az eljárások paramétereit (7.1) cím alatt foglalkozunk.

6.2 Az if..then utasítás

Az **if..then** a Pascal legegyszerűbb feltételes utasítása. Az **if** után általában tetszőleges kifejezést (vagy **boolean** típusú változót) írhatunk. A megkötés mindössze annyi, hogy a kifejezés kiértékelése után az eredmény **boolean** típusú legyen. A **then** után csak egyetlen utasítás állhat, ez azonban a **begin** és az **end** kulcsszavakkal képzett összetett utasítás is lehet. A megkötés végső soron tehát nem korlátoz, a **then** után tetszőlegesen hosszú programrészt írhat.

Az **if..then** utasítások gyakorlatilag tetszőlegesen mélységben egymásba skatulyázhatók, azaz a **then** után újabb **if** állhat stb.

A TUDOMÁNSZERVEZÉSI ÉS INFORMATIKAI INTÉZET

előzetes megbeszélés szerint díjmentes programmutatót tart (vidéken is) az általa forgalmazott oktatóprogramokból.
Horváth Zsuzsa 865-011/2663 mellék
vagy 813-197
Budapest, Pf. 454, 1372



6.3 Program tíz integer típusú adat sorba rendezésére

Végezetül a 17. ábrán — minden különösebb magyarázat nélkül —

bemutatjuk a feladatot megoldó program listáját. A tíz adatot egy tizelemű tömb elemeiként definiáltuk és kezeltük. (A program a közismert buborék algoritmus alapján végzi el a rendezést.) A futtatáskor keletkező screen a 18. ábrán látható.

BASIC-en nevelkedettek számára az ismeretanyag valószínűleg nehezebbnek bizonyul majd, mint a tiszta lappal indulóknak. A téma jelentősége miatt a szokásosnál több magyarázatot és egészen egyszerű példákat adunk.

7.1 Az eljárások paraméterei

Kiindulásként tételezzük fel, hogy rendelkezünk egy, már megírt eljárással, amely két — az eljárást hívó programból származó — számot összead, majd az összeget a hívó programnak visszaadja. Ennek az eljárásnak a listáját láthatjuk a 19. ábrán.

A dologban semmi furcsaság nem lenne, ha a változódeklarációt nem az eljárásnév utáni zárójelben, hanem a már megszokott helyen találnánk. Az ilyen módon deklarált változók a hívó program

7. Teljes erőbedobással az eljárásokért

Ahogy az a sorozat előző részében már említettük, egyre mélyebbre hatolunk a Pascal rejtelmeibe. Állításunk igazságát mi sem bizonyítja jobban mint az, amin az olvasónak most „át kell

rágnia magát”. Egyebek között ugyanis a Pascal egyik fő erősségével, a programkörnyezettől független eljárásokkal, azok készítésével és alkalmazásával fogunk foglalkozni. Megjegyzendő, hogy a

```

program szamsor;
{tömbök használata, globális és lokális változók,
if then}
var i:integer;
    tomb:array [1..10] of integer;
{globális változók: i,tomb}
procedure bevitel;
begin
    clrscr;writeln('Negyedik gyakorló programunk');
    writeln('Kérek 10 egész számot');
    for i:=1 to 10 do
        readln(tomb[i]);
end;
procedure rendez;
var j,temp:integer;
{lokális változók: j,temp}
begin
    for j:=1 to 9 do
        for i:=1 to 9 do
            if tomb[i]>tomb[i+1] then
                begin
                    temp:=tomb[i];
                    tomb[i]:=tomb[i+1];
                    tomb[i+1]:=temp;
                end
            end;
end;
procedure kivitel;
begin
    gotoxy(30,2);writeln('Az adatok rendezve');
    for i:=1 to 10 do
        begin
            gotoxy(30,2+i);
            writeln(tomb[i]);
        end
end;
begin {foprogram kezdet}
    bevitel;
    rendez;
    kivitel;
end.

```

17. ábra

```

program ujradeklaral;
var i:integer; {i első deklarációja}
procedure ujra;
var i:integer; {i ujradeklarálása}
begin
    i:=i;
    writeln('i ujradefiniált=',i);
end;
begin {foprogram}
    clrscr;
    i:=2;
    writeln('i eredeti=',i);
    ujra; {i új értéket kap}
    writeln('i eredeti=',i) {i értéke ismét a régi}
end.

```

15. ábra

16. ábra

```

i eredeti:=2
i ujradefiniált=1
i eredeti:=2
>

```

18. ábra

```

Negyedik gyakorló programunk
Kérek 10 egész számot      Az adatok rendezve
B7                          -12
1                            1
-12                          4
45                            6
22                            11
31                            20
20                            22
4                              31
6                              45
1                              87
>

```



és a hívott eljárás közötti adatcserét szolgálják. Feladatuk más, mint egy program vagy eljárás már ismert módon, az ún. deklarációs részben definiált változóinak. E különbség miatt a Pascal más elnevezést használt megjelölésükre: formális paramétereknek nevezzük őket.

Közülük azok, amelyek a kezdő zárójeltől a **var** alapjelig vannak felsorolva, a bemeneti paraméterek, azaz értéküket a hívó programból kapják (példánkban az **a** és **b**). A **var** alapjel után lévők az eljárás kimeneti paraméterei, ezek az eljárás végrehajtása után a hívó programnak adnak át értéket (példánkban ilyen az **összeg**).

Már most — az eljárás hívásának bemutatás előtt — megemlítjük, hogy a formális paraméterként definiált változókra ugyanazok a szabályok érvényesek, mint a lokális változókra; mivel hatáskörük csak az adott és az abból hívott eljárásokra terjed ki, a változónevek kiválasztására speciális korlátozás nincs. (Közömbös, hogy az eljárás paramétereit a program más részében változóként vagy paraméterként alkalmaztuk-e vagy sem.) Egyébként éppen ez az a tény, ami az eljárásnak a programkörnyezettől való teljes függetlenségét biztosítja!

A kérdés most már csak az, hogyan kapnak a formális paraméterek értéket? Ennek megoldása sem

bonyolult: az eljárás hívásakor az eljárásnév után zárójelben, egymástól vesszővel elválasztva fel kell sorolni az eljárásnak átadni, illetve onnan átvenni kívánt adatokat, azaz a szám- vagy szövegkonstansokat, változókat, kifejezéseket. A felsorolásban szereplő elemek elnevezése: aktuális paraméterek.

A 19. ábrán bemutatott eljárást használó program listája látható a 20. ábrán.

A program a billentyűzetről beolvasson két számot (**op1** és **op2**). Ezután hívja az **osszead** eljárást. A paraméterátadás a következők:

- **op1** értéke átadódik **a**-nak,
- **op2** értéke átadódik **b**-nek,
- az eljárás elvégzi az összeadást, az eredmény az **összeg** változóba kerül,
- az **összeg** értéke átadódik **szumma**-nak.

A bemutatott példából két nyilvánvaló szabályra derül fény:

- a formális és aktuális paraméterek száma azonos legyen,
- a formális és aktuális paraméterek típusa azonos vagy legalábbis kompatibilis legyen.

A szabályok magyarázata igen egyszerű. A program lefordításakor a fordító helyet foglal el a tárban a formális paraméterek listája alapján. Ha ezekre a helyekre nem illeszkednek a program futása közben az aktuális paraméterek, ez hibához vezet. A paraméterek és a

tárhelyek illeszkedésének feltétele: azonos számú és azonos helyfoglalású (kompatibilis) paraméter.

A típuskompatibilitást a különböző Pascal reprezentációk különféleképpen kezelik. Általános útmutatásként az fogadható el, hogy a különböző típusok akkor kompatibilisek, ha azonos alaptípusra vezethetők vissza. Például várhatólag kompatibilis egy húszkarakteres sztring és egy húszkelemű karaktertömb. Mindezért legcélszerűbb, ha a birtokunkban levő fordítók kipróbáljuk a kompatibilisnek vélelmezhető típusokkal. Nekünk egy bombabiztos javaslatunk van: egyelőre ne kompatibilisnek vélt, hanem azonos típusú paraméterekkel dolgozzunk!

Az érdekesség és a tanulság kedvéért **szumma** nevű programunk egy másik változatát is bemutatjuk. Ez a program ugyanazokat a neveket használja a programban és az eljárásban, sőt formális és aktuális paraméterként is. A program listája a 21. ábrán látható.

A program a sok adat azonos neve mellett is működik. Ennek magyarázatát, indoklását az olvasóira bizzuk! Ha valakinek nehézségei támadnának, javasoljuk, hogy olvassa végig még egyszer a globális és lokális változókról leírtakat (6.1 rész).

N. I.

```
procedure osszead(op1,op2:real;var szumma:real);
begin
  szumma:=op1+op2;
end;
```

19. ábra

```
program szumma;
var op1,op2,szumma:real;
procedure osszead(a,b:real;var osszeg:real);
begin
  osszeg:=a+b;
end;
begin
  write('1. osszeadando:');readln(op1);
  write('2. osszeadando:');readln(op2);
  osszead(op1,op2,szumma);
  writeln('Az osszeg: ',szumma)
end.
```

20. ábra

```
p.ogram szumma;
var op1,op2,szumma:real;
procedure osszead(op1,op2:real;var szumma:real);
begin
  szumma:=op1+op2;
end;
begin
  clrscr;
  write('1. osszeadando:');readln(op1);
  write('2. osszeadando:');readln(op2);
  osszead(op1,op2,szumma);
  writeln('Az osszeg: ',szumma)
end.
```

21. ábra



Commodore Plus/4

**Program-
ajánlat**



Tettre kész mindenés

A professzionális kivitelű program segítséget nyújt levelek, iskolai segédletek (feladatlapok, tesztlapok stb.) és tetszőleges írásos dokumentumok szerkesztésére. Az elkészítendő nyomtatványok sokszínűségét elősegíti az, hogy a program képes valódi grafikus ábrák szerkesztésére és a szövegbe illesztésére, valamint lehetőséget ad magyar ékezetes, cirill, görög és „indexbe írt” karakterkészletek használatára.

Az írat elkészítéséhez egy százsoros, soronként 50 karakteres szerkesztőterület áll rendelkezésre. Ebből a képernyőn huszonegy sor és soronként 35 karakter látható. A képernyő többi részét a fent és baloldalt állandóan jelen lévő ikonmenü tölti ki. A felső menü általános célú funkciók kiválasztására szolgál, az oldalsó a grafika szerkesztését segíti. A kurzort vezérlő billentyűkkel természetesen a szöveg bármelyik része hozzáférhető. A szerkesztőterület akármelyik részén tetszés szerint helyezhetünk el ábrákat, szöveget.

A számítógép képernyőszerkesztőjéhez hasonlóan használhatók az INST/DEL és a RETURN billentyűk. A HOME a képernyő, a CLEAR az írat kezdőpozíciójára állítja a kurzort. A billentyűk önmagukban kis, SHIFT-tel együtt nagy, a C=-ral együtt pe-

dig ékezetes betűket jelenítenek meg. További szerkesztési funkciókat az ikonmenü segítségével végezhetünk el.

A megfelelő ikont — piktogramot — a kurzorral választhatjuk ki, amely a menüben mint nyíl jelenik meg. A menüelem aktivizálása a SHIFT billentyű lenyomásával érhető el.

A szöveg írása közben bármikor válthatunk karakterkészletet. Az F1,

F2, F3 és a HELP billentyűk rendre a magyar, a cirill, a görög és az index betűkészlet váltóiként alkalmazhatók.

Gyakran lehet szükség arra, hogy nyomtatáskor egy szövegrészt a papír közepére vagy jobb szélére helyezzünk. Ezt valósíthatjuk meg a CONTROL billentyű használatával. Egyszeri lenyomás hatására a képernyő egy sora a papíron középre, ismételt lenyomásakor a papír jobb oldalára kerül.

Billentyűzet megfeleltetési táblázat a cirill betűk használatához:

а-а	з-з	п-п	ч-ч
б-б	и-и	р-р	ш-ш
в-в	я-я	с-с	щ-щ
г-г	к-к	т-т	м-м
д-д	л-л	у-у	ы-ы
е-е	м-м	ф-ф	о-о
ё-ё	н-н	х-х	ю-ю
ж-ж	о-о	ц-ц	я-я

Billentyűzet megfeleltetési táblázat a görög betűk használatához:

α-α	ι-ι	ρ-ρ
β-β	κ-κ	σ-σ
γ-γ	λ-λ	τ-τ
δ-δ	μ-μ	υ-υ
ε-ε	ν-ν	φ-φ
ζ-ζ	ξ-ξ	χ-χ
η-η	ο-ο	ψ-ψ
θ-θ	π-π	ω-ω



Egy kép a program képernyőjéről:



Ha a kurzor egy grafikus kép területéhez ér, akkor a képből az aktuális ceruzaheggyel ellátott nyíl jelenik meg. Ez mindaddig „él”, amíg a kép területét el nem hagyjuk. A grafikus menübe a grafikus területből annak bal oldali választófalán át juthatunk be. A funkciót az általános menüvel kapcsolatban már leírtakhoz hasonlóan választhatjuk ki.

A programot írták: Petrov Ferenc és Petrov Péter
 Forgalmazza a Bács-Kiskun Megyei Pedagógiai Intézet
 6001 Kecskemét, Katona József tér 8. Pf. 180.

Tempótartás kopogással

A Magazin tavalyi 6., 7. és 10. számaiban már jelentek meg zenei képességeket vizsgáló rövid programok C64-re. Ennek a gépnek a hanggenerátora és belső órája lehetővé teszi, hogy

```

5 DIM X(16),Y(16):READ H:FOR I=1 TO H
10 READ Y(I):O=O+Y(I):NEXT
15 O=O-Y(I-1)
20 DATA 7,6,2,4,4,4,4,B
25 S=54272:FOR I=0 TO 24:POKE S+I,O:NEXT
30 POKE 53281,1:POKE S+24,15:POKE S+5,24
35 M=INT(RND(1)*7)+B
40 PRINT CHR$(147):FOR I=1 TO H:TI$="000000"
45 POKE S+4,129:POKE S+1,50
50 IF TI<Y(I)*M THEN 50
55 POKE S+4,128:NEXT
60 PRINT "MI LEGYEN A FOLYTATAS?":PRINT
65 PRINT "1. UJBOL MEGHALLGATOD A RITMUST"
70 PRINT "2. VISSZAKOPOGD":POKE 198,0
75 GET R$:IF R$="" THEN 75
80 IF R$="1" THEN 40
85 PRINT CHR$(147)"ELKEZDHEDED"
90 POKE 198,0:FOR I=1 TO H:TI$="000000"
95 GET R$:IF R$="" THEN 95
100 IF I=1 THEN PRINT CHR$(147):NEXT I
105 X(I-1)=TI:NEXT
110 FOR I=1 TO H:SU=SU+X(I):NEXT:K=SU/(M*O)
115 IF K<=1 THEN SZ$="GYORSABB":GOTO 125
120 SZ$="LASSABB"
125 PRINT "A TE KOPOGASOD":
130 PRINT ABS(INT(10000*(1-K)+.5)/100):
135 PRINT "%-KAL VOLT "
140 PRINT SZ$ "AZ EREDETINEL":PRINT
145 PRINT "AZ EGYES UTESEK ELTERESEI"
150 PRINT "(A TE TEMPOD SZERINT) %-ban:"
155 FOR I=1 TO H-1:A=100*X(I)/(Y(I)*M*K)
160 B=INT(10*A+.5)/10-100
165 C=INT(100*B+.5)/100:PRINTTAB(13)C
170 NEXT:END
    
```

olyan zenei képességekről szerezzünk pontos adatokat, amiket korábban a legjobb felül szakemberek is csak szubjektív értékkelhettek. Természetesen nem kizárt, hogy a zenét nem tanulók is alávessék magukat a megmértetésnek. Az ismertetendő téma az életben sok helyütt felbukkan. Hiánya vagy megléte befolyásolhatja hétköznapjainkat. A téma nem más, mint a tempó-tartás.

A program egy közismert ritmust, a magyar Himnusz kezdő-sorát szólaltatja meg (40–55-ös sorok). A megszólalás tempóját a program határozza meg (35-ös sor). Ezt kell visszakopogni valamelyik billentyűn (90–110-es sorok). Ezután a program úgy méri a tempó tartását, hogy összehasonlítja a hangoztatás és a kopogás idejét (125–140-es sorok). Ez azonban nem egészen korrekt, mivel az egyes hibák szerencsés esetben kompenzálhatják is egymást. Ezért az egyes ütések hosszát is összehasonlítja az ideállal (145–170-es sorok), úgy, mintha az etalon is a visszakopogás átlagos tempójában szólt volna meg (155-ös sor). Zenét tanulóknak hasznos lenne az is, ha az egyes ritmus-képletek pontosságát önmagukban is vizsgálnánk. Ez azonban lényegesen bonyolítaná a programot. E két vizsgálat akkor nyújt igazán reális képet, ha a programot többször futtatjuk egymás után, s így többféle tempóban halljuk a ritmust.

A kiértékeléshez az alábbi megközelítő adatokat lehet figyelembe venni. Ha a kopogás és hangzatás időtartama közötti eltérés 0-2 százalék között van, az kiváló eredmény; 2-5 százalék közötti érték jó, az 5-10 százalék már rossz eredménynek számít. 10 százalék fölött bizonyára más tempót akart a felhasználó megvalósítani, ezért ezt nem lehet még rosszabbnak minősíteni.

Ha az egyes ütközések eltérése 0-6 százalék között van, az kiváló; 6-15 százalék között jó, 15-30 százalék között rossz eredménynek minősíthető. 30 százalék fölött valószínűleg téves volt a ritmus értelmezése.

A program lehetőséget nyújt más ritmusok bevitelére is. Ilyenkor a 20-as sorban elsőként azt kell közölni, hogy hány hangból áll a ritmus; nálunk hétből. Utána a ritmust relatív értékekkel kell megadni. Célzerű a negyed értékű hanghoz a 4-es számot rendelni. Ekkor a nyolcad a 2-nek, a tizenhatod az 1-nek stb. felel meg. Ha a negyed érték a 6-os számot kapja, akkor triola és nyolcad egymás mellett is megszólaltatható.

Kalmár Gyula



Az írógéptől a számítógépig

Akik fellapozták az 1987/88-as tanévre szóló Fővárosi Pályaválasztási Tanácsadó c. kiadványt, még hiába keresték benne az Arendás közti — Újpalota — Közlekedési és Közgazdasági Szakközépiskola címét. Pedig ebben az iskolában már tavaly szeptember óta folyik a leendő közlekedés-gépészeti technikusok és közgazdasági szakemberek képzése.

a pénzt a Budapest Főváros Tanácsa V. B. művelődésügyi főosztálya, valamint a Közlekedési Minisztérium közreműködésével a közlekedési vállalatok — pl. BKV, Hungarocamion, Volán Tefu stb. — biztosították. Persze ez utóbbiak saját jól felfogott érdekekből tették ezt. Ezeknél a vállalatoknál a régi könyvelőgépeket már lecserélték IBM PC XT vagy AT kategóriájú gé-



Eddig ez nem jelent semmi szenzációt, hiszen bizonyára nemcsak ebben az intézményben kezdik meg első ízben a szakemberképzést. Érdeklődésünk inkább arra irányult, hogy hogyan tudja egy újonnan induló iskola a napjainkban oly fontos, a jövőben pedig egyenesen nélkülözhetetlen számítástechnika oktatásának feltételeit megteremteni. Erről Donázy István igazgatótól kaptunk tájékoztatást.

Az épületet eredetileg általános iskolának szánták, de a demográfiai hullám ismert alakulása miatt szükségessé vált középfokú intézménnyé való átalakítása. Ez rendkívül sok pénzt emésztett fel, ezért nem is gondolhattak arra, hogy önerőből komolyabb számítástechnika-oktatási hátteret teremtsenek. A gépbszerzésekhez

pekre. A jelenlegi oktatási rendszerben az iskolák többnyire csak követni próbálják a számítástechnikai igényeket. Az iskolavezetőség itt előrelátó volt, és a régi latin közmondás szellemében csakugyan az élet-

nek tanítanak. Hogyan? A számítástechnika oktatásához két terem áll rendelkezésre.

Sikerült beszerezniük tíz C64-et, a TII kedvezményes áron biztosított tíz IBM PC XT és egy AT típusú gépet, melyekhez winchesterek, színes monitorok és nyomtatók is rendelkezésre állnak. A C64-es gépeket arra használják, hogy azok is megismerkedhessenek a számítógéppel, akiknek otthon nincs gépük, tehát ezek amolyan BASIC-kóstolásra szolgálnak.

Amikor Kun Emese tanárnő vezetésével elsajátították az alapokat, akkor jön az igazi munka a PC-ken. Az említett közlekedé-

si vállalatok a tervek szerint biztosítják az iskola részére a saját, élesben futó rendszereiket, amelyeket a hallgatók tananyagként sajátítanak el. Azok a hallgatók, akik tanulmányi szerződést kötnek valamelyik vállalattal, így „kész” alkalmazási szakemberként képesek már az első munkanaptól dolgozni, hiszen megtanulták a leckét.

Az átlagosnál érdeklődőbb hallgatók számára a tanterven felül a számítástechnikai szakkörök is rendelkezésre állnak. Az iskola várja az olyan pedagógusok jelentkezését, akik kedvet éreznek a számítástechnika oktatásához.

Sz. S.





Írni nehéz...

Kettőt egy csapásra

A mellékelt listán lévő programot kettős céllal bocsátjuk közre. Részben segíteni kívánunk azoknak (lásd a későbbieket), akik a programot a tanulásban vagy a tanításban alkalmazni kívánják, másrészt pedig gondolkodásra szeretnénk készíteni azokat, akik kedvet éreznek egy program logikájának kibogozásához. A program C Plus/4-es gépre készült; természetesen a C16-tulajdonosok minden változtatás nélkül használhatják.

A program a nem összetett szavakat szótagolja. Sajnos az ékezetes betűket nem ismeri. Ezek kezelése is megoldható lett volna ugyan, de ebben az esetben — céllal ellentétben — éppen az egyszerűséget vesztettük volna el. A magyar nyelv elválasztási szabályait az ékezetes vagy ékezet nélküli magánhangzók használata egyébként sem befolyásolja.

Mint ahogyan ezt az általános iskolák tanárai és a korosztályba tartozó gyerekek szülei egyaránt tudják, a kisebbek egy részének nagy gondot okoz a szótagolás, vagy ami ezzel egyenértékű, a szavak elválasztása. Eppen ezért a program kiválóan alkalmas a kicsiknek gyakorlásra, illetve a felnőtteknek a gyakoroltatásra.

Hogy másoknak, a BASIC-ben már többé-kevésbé járatosoknak is hasznosat nyújtsunk, azt javasoljuk, próbálkozzanak meg a program működésének megfejtesével. Ez a feladat a program látszólagos egyszerűsége ellenére sok fejtevést fog okozni. Azok azonban, akik a feladattal eredményesen megbirkóznak, sikerélményre számíthatnak. Az ő dolgukon könnyítendő hívjuk fel a figyelmet a 150-es sorra: ehhez úgy férhetnek hozzá, ha a sorban lévő REM utasítást a programból kiirtják.

Horváthné Tökei Zsuzsanna

```
10 DIM#(21) : DATA E, I, O, U, CS, DZ, GY, LY, NY, SZ, TY, ZS
20 DATA CS, DDZ, GGY, LLY, NN, SSZ, TTY, ZZS
30 FOR I=1 TO 1: READ#(1) : NEXT I
40 C$="" : SCREEN PRINT INPUT " A TAGOLANDÓ SZÓ:";#
50 FOR I=1 TO LEN(C$)
60 FOR J=1 TO 5: IFB$(J)=MID$(C$, I, 1) THEN C#=C$+"0" : GOTO140
70 NEXT J : C$=C$+"1"
80 FOR J=6 TO 21
90 IFB$(J)=MID$(C$, I, 2) THEN C#=C$+"2" : I=I+1 : GOTO140
100 IFB$(J)=MID$(C$, I, 3) THEN C#=C$+"22" : I=I+2 : GOTO140
110 IFMID$(C$, I, 3)="DZS" THEN C#=C$+"13" : I=I+2 : GOTO140
120 IFMID$(C$, I, 4)!="DDZS" THEN C#=C$+"133" : I=I+3 : GOTO140
130 NEXT J
140 NEXT I : C$=C$+"#"
150 PRINT : REM PRINT#C$
160 P1=1 : I=1
170 IFMID$(C$, I, 1)!="0" THEN190
180 I=I+1 : GOTO170
190 IFMID$(C$, I, 1)="#" THEN200
200 I=I+1 : IFMID$(C$, I, 1)!="0" THEN220
210 GOTO190
220 P=VAL(MID$(C$, I-1, 1)) : P2=1-P
230 PRINTMID$(C$, P1, P2-P1) :
240 IFMID$(C$, P2, 2)!="22" THENPRINTMID$(C$, P2+1, 1) : GOTO260
250 IFMID$(C$, P2+1, 2)!="33" THENPRINTMID$(C$, P2+1, 2) :
260 PRINT " : "
270 P1=P2 : GOTO190
280 PRINTMID$(C$, P2, 255)
290 CHR$(10, 23) : NYOMJ LE EGY BILLENTVUT1"
300 GETKEY# : RESTORE : GOTO300
```

Sorozatunk elsősorban középiskolásoknak szánjuk, de reméljük, hogy minden olvasónknak tanulási lehetőséget és szórakozást nyújt.

A feladatok a Nemes Tihámé országos számítástechnikai verseny színvonalának felelnek meg. Minden esetben olyat választunk, amely röviden, gyorsan megoldható, de megoldásához ötletre van szükség. A megoldást mindig a következő számban közöljük.

Mivel változatosságra törekszünk, különböző programozási nyelveket használunk. Az is előfordul, hogy egy feladatra több programnyelven is közzé tesszük a megoldást, ezzel is elősegítve az ismeretszerzést.

A szerkesztőség várja az olvasók, a versenyzők leveleit. A legötletesebb program beküldőjét könyvtulajdonnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

10. feladat:

Labirintusrajzolás

Írjon programot, amely véletlen labirintust rajzol! A labirintusnak a következő tulajdonságai legyenek:

1. Egy be- és egy kijáratra legyen.

2. A bejáratról a kijáratig egy és csak egy út vezessen.

3. A bejáratról a labirintus bármely pontjára el lehessen jutni.

Megoldás

Véletlen labirintus rajzolására számtalan módszer képzelhető el. A legegyszerűbb az lenne, ha a képernyőre véletlenszerűen vonalak húzgalnának. Ennek az ötletnek a továbbfejlesztése az, hogy a vonalakat nem teljes összességükben húzzuk, hanem a kezdő- és a végpont koordinátáira valamilyen megkötést teszünk. Ha a vonalakat csak páros koordinátájú pontok között húzzuk, akkor egész szép labirintusokat kaphatunk.

Természetesen az ilyen labirintusok nem teljesítik a feladat második és harmadik feltételét. Szinte biztosan tartalmaznak zárt, minden oldalról fállal körülhatárolt részeket, a bejáratról a kijáratig vezető út létezése pedig igencsak kétséges. Ha ezeket a hibákat valahogyan megszüntetnénk, máris megkapnánk a megoldást.

Egy másik lehetséges megoldáshoz juthatunk a következő gondolatmenettel.

Induljunk ki a még felosztatlan táblából! Véletlenszerűen választott helyen osszuk ketté a téglala-

pot egy olyan fallal, amelyen pontosan egy átjáró van (lásd az ábrákat).

Ismételjük meg ezt az eljárást a jobb és a bal oldali téglalapra!

Most végezzük el ugyanezt a keletkezett négy téglalapra! Folytassuk az eljárást mindaddig, míg a téglalapok tovább már értelmetlenül nem oszthatók!

Az így kapott labirintus a kívánalmaknak minden szempontból megfelel, bármely pontból bármely más pontba pontosan egy út vezet.

Ezenkívül okozkodhatunk a következőképpen is. Ha megvizsgálunk egy, a feladat feltételeinek megfelelő labirintust, láthatjuk, hogy az két összefüggő falrendszerből (fából) áll. A két falrendszer között halad a labirintuson átvezető út. Ha tudunk olyan programot írni, amely ezt a két véletlenszerű fát megrajzolja, akkor lényegében már meg is oldottuk a feladatot. Ilyen fát a következőképpen rajzolhatunk.

Jegyezzük fel egy naplóba a fa meglévő ágainak azon pontjait, ahonnan további ágakat lehet növeszteni. Minden lépésben válasszunk ki véletlenszerűen egy-egy pontot. Vizsgáljuk meg, hogy a négy lehetséges irány közül valamelyikben továbbléphetünk-e. Ha nem, töröljük a pontot a naplóból — egyébként pedig az egyik lehetséges irányba növezzük tovább az ágat, és a kapott új pontot jegyezzük be a naplóba. Amikor elfogytak a folytatható ágak, a labirintus rajzolása befejeződött.

A felsorolt módszerek mindegyike egyaránt alkal-

FELADATOK — MEGOLDÁSOK

mas a feladat megoldására, de talán a másodikát egy kicsivel könnyebb beprogramozni.

A közölt program is ezt használja. A Turbo Pascal 4.0 nyelven megírt program IBM PC-n tetszőleges grafikus adapterrel működtethető.

A program fő részét Xfelez és Yfelez eljárások alkotják. Ezek osztják ketté a téglalapot vízszintes, illetve függőleges irányban. A két szubrutin olyannyira hasonló, hogy működésük egységesen tárgyalható.

Az eljárás paraméterként megkapja a kettéosztandó téglalap adatait: a bal felső sarok koordinátáit, a téglalap szélességét és hosszúságát. Amennyiben a téglalap szélesebb vagy magasabb

mint az SZ konstansban tárolt folyosószelesség, az osztófal (oszt) és a falon átvezető átjáró (lyuk) helyének kiszámítása következik. A következők két sor a fal megrajzolását látja el.

A fal két oldalán keletkeztetett téglalapok további felosztását Xfelez vagy Yfelez végzi KOMPL konstans értékétől függően eltérő valószínűséggel. Ezzel a labirintus bonyolultságát lehet szabályozni.

A főprogram feladata

már csak a grafikus mód beállítása, a külső keret, a be- és kijárat megrajzolása. A labirintus rajzolását a főprogramban meghívott Xfelez végzi el. Végül még szerepel egy ellenőrzés: a FloodFill utasítás kiszínezi a labirintust. Ha az egész képernyőt betölti a szín, az jelzi, hogy a labirintus bármely pontja elérhető a bejárattól indulva.

A következő feladat témája bizonyára sokak előtt ismeretes.

11. feladat:

Egér a labirintusban 1.

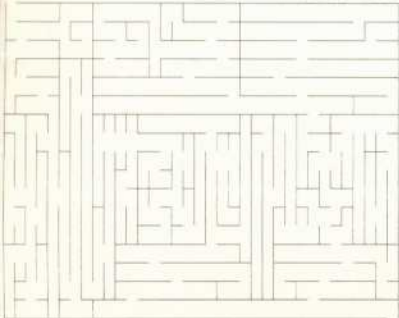
Tegyük az egeret a labirintus bejáratához, a sajtot pedig a kijáratához. Megtalálja-e az egér a sajtot? Őn, mint az egér jó barátja, elhatározta, hogy segít neki. Hogyan? Irjon erre programot! — Tegyük fel, hogy a labirintus térképe nem ismert, az egér csak az öt közvetlenül körülzáró falakat látja.

Pintér Gábor

A tábla az első kettéosztás után



A kész labirintus



```

(-----)
(                               )
(   Labirintus rajzolo program )
(                               )
(                               )
(   Pinter Gabor                )
(                               )
(                               )
(   1988.10.06.                 )
(                               )
(-----)

program labirintus;

uses
  Graph;

const
  SZ:=20; { Folyoso szelesseg }
  KOMPL:=8; { Labirintus bonyolultsag }

var
  MaxX, MaxY : integer;
  Ch          : char;
  Gd, Gm      : integer;

procedure Yfelez(
  x,y,dx,dy : integer); forwards;

{ Fuggolegesen kettéosztja a tablát }
procedure Xfelez;
  x,y, { Bal felső sarok koordinatai }
  dx,dy { Tabla merete }
  : integer;

var
  { Az osztófal helyzete: }
  oszt : integer;
  { Az osztófalra a lyuk helye: }
  lyuk : integer;

begin
  { Kell-e tovább osztani? }
  if (dx>=2*SZ) and (dy>=2*SZ)
  then begin
    { Fal rajzolásai }
    oszt:=x*SZ+Random(dx div SZ - 1)+SZ;
    lyuk:=y*SZ+Random(dy div SZ);
    Line(oszt,y,oszt,lyuk);
    Line(oszt,lyuk,oszt,y*dy);
    { A fal két oldalának levo
      tablak továbbosztasai }
    if Random(10)>KOMPL
    then
      Xfelez(x,y,oszt-x,dy)
    else
      Xfelez(x,y,oszt+x,dy);
    if Random(10)>KOMPL
    then
      Xfelez(oszt,y,x+dx-oszt,dy)
    else
      Yfelez(oszt,y,x+dx-oszt,dy)
    end (if);
  end (proc);

```

```

{ Vízszintesen kettéosztja a tablát }
procedure Yfelez(
  x,y, { Bal felső sarok koordinatai }
  dx,dy { Tabla merete }
  : integer);

var
  { Az osztófal helyzete: }
  oszt : integer;
  { Az osztófalra a lyuk helye: }
  lyuk : integer;

begin
  { Kell-e tovább osztani? }
  if (dx>=2*SZ) and (dy>=2*SZ)
  then begin
    { Fal rajzolásai }
    oszt:=y*SZ+Random(dy div SZ - 1)+SZ;
    lyuk:=x*SZ+Random(dx div SZ);
    Line(x,oszt,lyuk,oszt);
    Line(lyuk*SZ,oszt,x+dx,oszt);
    { A fal két oldalának levo
      tablak továbbosztasai }
    if Random(10)>KOMPL
    then
      Xfelez(x,y,dx,oszt-y)
    else
      Yfelez(x,y,dx,oszt-y);
    if Random(10)>KOMPL
    then
      Xfelez(x,oszt,dx,y+dy-oszt)
    else
      Yfelez(x,oszt,dx,y+dy-oszt)
    end (if);
  end (proc);

begin
  Randomize;
  repeat
    { Grafikus mod }
    Gd := Detect;
    IniGraph(Gd, Gm, '');
    if GraphResult < grOk then
      Halt(1);

    { Keret a be- es kijáratral }
    MaxX:=SZ*(GetMaxX div SZ);
    MaxY:=SZ*(GetMaxY div SZ);
    Line(0,0,0,MaxY);
    Line(0,MaxX,MaxX,SZ,MaxY);
    Line(MaxX,0,SZ,0);
    { Labirintus rajzolás }
    Xfelez(0,0,MaxX,MaxY);

    { Vegs }
    Readln(ch);

  CloseGraph;
  until ch=#13
end.

```

Diagnózis?

PcVÍRUS

Terápia?

PcHIGI

```
C:\>dir o d
. Volume n o
. Direct I uAI
ACAD-NCoi BEX
AAAA-NCI rXEX
AAAA-NC25XFi
Virus??
Ne_ásoljon!
1 -
u 4 12
h:3C i9 - 988
Ju 12-29-1988 VIR
T8:36:30.611-B -80 2:01
iDIR>L569331-01-80 12:01a
Cap <D132244.S11-01-80 12:04a
r<(s317164981-01-80 12:05a
C:\> fehU)12-29-16881-80US12:06a
C:\>captu 1T8:34:23.019881bytesifreea
```

Bővebb tájékoztatóval
és programbemutatóval
rendelkezésre áll a

SOFTinvest
SZOFTVERKERESKEDELMI
ÉS FEJLESZTÉSI BETÉTI TÁRSULÁS
1391 Budapest, Pf. 218 Tel.:119 - 067, 328 - 769

Pályázati felhívás

Idén nyáron ismét megrendezik Novoszibirszkben a *Fiatal Programozók Nyári Táborát*, amelyen a *Neumann János Számítógéptudományi Társaság* Magyarországról összesen négy fő részvételét biztosítja. A tábor munkájában az a négy fő vehet részt — az *NJSZT* költségén —, aki az alábbi feladatra a legjobb megoldást küldi be.

FELADAT: A felhasználó által adott n számú, mindkét oldalán fehér kartyalapot kiterítünk az asztra. A kartyákat felfelé néző oldalukon 1-től n-ig sorszámozzuk, majd megfordítjuk és összekeverjük. Az újabb terítést követően a kartyák fehéren maradt oldalát is sorszámmal látjuk el. Ha kész, ismét megkeverjük a lapokat, de közben meg is fordítunk néhányat. Végül az asztra helyezett kartyák adatait — külön a felfelé, külön a lefelé néző oldalukon szereplő számokat — balról jobbra sorrendben feljegyezzük.

BEKÜLDENDŐ: az a *BASIC* nyelven írt, printerrel készített programlista és az az algoritmus (blokkdiagram), amely a fenti adatok alapján meghatározza, mely kartyákat kell megfordítanunk ahhoz, hogy a kartyák felfelé néző oldalán az 1-től n-ig terjedő számok mindegyike pontosan egyszer szerepeljen.

SZAKÉRTŐJE ÖN A SZÁMÍTÁSTECHNIKÁNAK? Kérje felvételét Társaságunk névjegyzékébe!

A Minisztertanács 40/1988. (V. 31.) MT határozata kiterjesztette a szakértői tevékenységet a szakismeretek közlésére, alkalmazására a vállalati innováció területén — a szoftver és hardver megkülönböztetése nélkül —, honorálására pedig a d ö k e d v e z m é n y t biztosított.

A SZÁMÍTÁSTECHNIKA TERÜLETÉN SZAKÉRTŐI TEVÉKENYSÉGRE VAN SZÜKSÉGE?

Vállaljuk:

- a szakmai szakértői névjegyzékbe bejelentkező szakembereket a szakértelmüknek megfelelő munkákra ajánljuk, róluk referenciát adunk,
- ezt a névjegyzéket hozzáférhetővé tesszük az innováció iránt érdeklődő vállalatok számára.

Közreműködünk a szakértői megbízás létrehozásának adminisztratív és pénzügyi lebonyolításában.

Fontos szempont az algoritmus, illetve program elbírálásánál, hogy gyors legyen, és minél kevesebb lépésből álljon.

Beküldési határidő: 1989. április 15.
Cím: *Mikroszámítógép Magazin Szerkesztősége*
1371 Budapest, Pf. 433.

Tájékoztatást ad
Aranyosné Varga Gabriella,
telefon: 329-349, 329-390.
Címünk: **Neumann János**
Számítógéptudományi
Társaság
1054 Budapest, Báthori u. 16.

NJSZT

A nagy sikerű, The last ninja című játék folytatásáról van szó. Ha lehet, még egy fokkal jobb animációja és zenéje, valamint szebb kivitelű grafikája van, mint elődjének. Történetünk New Yorkban játszódik, amint az a grafikából is kiderül. S még valami, amiben bizonyosak lehetünk: elbizakodottságra semmi okunk a leírás biztonságában sem, mert az ördögösségekkel újfent meg kell küzdenünk, ahányszor csak birokra kelünk.

The Central Park

Induláskor a Central Park egyik térszíni pódiumán állunk. Menjünk be a függöny mögötti terembe és vegyük meg a rejtejt levezető lépcső kapcsolóját őrző illetőt. Nyomjuk meg a szemközti falon található kapcsolót — ez beléptünkkor villogással hívja fel magára a figyelmet, mint minden fontos tárgy, amit össze kell szednünk vagy használnunk kell a játék során —, és térjünk vissza a nagydob mellé, ahol közben megnyílt a levezető lépcső ajtaja.

Essünk le — emberünk ruganyosan földet ér. Szedjük fel a bal alsó sarokban lévő kis kulcsot és távozzunk a nyitott ajtón. (A leessé négyoszóba való visszalépéskor visszakerülünk a pódiumra.)

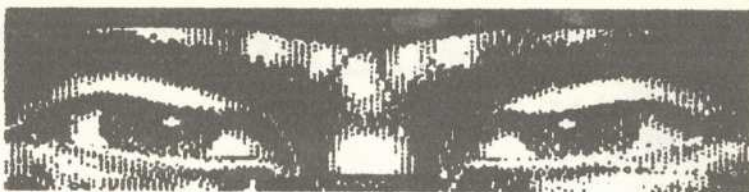
Kilépéskor a parkban találjuk magunkat, és a szemben látható kapu kulcsa van a zsebünkben. De még ne menjünk tovább!

Utasítsuk rendre a bal alsó sarokban mázskáló rendőrt — ezt ő is megteszi velünk, ha túl közel kerülünk hozzá —, és távozzunk. Ezen a pályán az út mellett található díszkockából kivehetjük kedvenc dobócsillagunkat. Tovább haladva felfelé, egy rosszindulatú szonglör próbál kilőni minket. Ugráljuk át a lövedékeket. A következő terembe lépve egy újság hívja fel magára a figyelmet. Vegyük fel, és másszunk fel a aszrakra. Balra haladva újabb fegyverünket, egy botot látunk. Fáradjunk el érte és térjünk vissza a kertkapuhoz. Takarékoskodjunk a dobócsillaggal, mert nagy szükségünk lesz rá máshol is! A kapunál elmehtünk felfelé egy falas hamburgerért és onnan jobbra egy WC-hez, amit egy zord vécsé néni őriz.

Nyissuk ki a kaput és menjünk be. Ugorjunk be a leszáguldó csónakba, majd onnan a másik partra. A méhek elől szaladjunk a bal felső sarokba, és ugorjunk egy nagyot, ugyanis a látszat ellenére ott egy sziget található. A szigeten lökjük el a csónakot a bottal, majd térjünk vissza szűros barátainkhoz. Most balra távozzunk, és ismételjünk meg az előző pályán tanult csónakos mutatványt. A túlparton induljunk lefelé.

The street

Az utcára lépve induljunk felfelé, a következő pálya után menjünk át a lámpás zebrán, lehetőleg zöldnél, különben elüt egy jökepű motoros. A lámpa után menjünk balra. Az itt található lelkes amatőrt vegyük meg és rúgjuk be a szemközti ajtót. Vegyük le a falon lévő kardot. Visszafelé menet ne a lámpás, hanem a sima zebrán keljünk át, és óvakodva a virágcserepektől, menjünk tovább. Vegyük fel a hamburgert,



majd a következő pályán a csavargó előtti üveget és menjünk át a zebrán. Tovább haladva intézzük el a két botos urat és kutasuk át a második bódéját. Egy fura szerzőmot találunk, ami nagyon fontos. Térjünk vissza a rendőrig (a hamburgeres kocsi előtti pálya), ahol egy zebra vezet felfelé. Menjünk át rajta. Innen már csak három pályányi a kiindulópontunk. Menjünk újra felfelé, de most a rendőr mellett forduljunk jobbra. Itt találunk egy csatornafedőt, amit a szerzőmással felnyithatunk. Ereszkedjünk alá.

The sewers

Induljunk jobbra, majd onnan a második pályán vegyük fel a földön lévő szerzőmot. Most folytassuk utunkat felfelé az előző pályán, a nyílást átugorva. Ugorjunk át ismét és várjuk meg, amíg a guszpusos pók elgaloppozik, és siessünk utána. Innen már csak háromteremnyire van a kiindulópont. Ott emeljük fel a vasrácsot a szerzőmással, és szálljunk alá. A leérkezést követő pályán menjünk be a középső ajtón. Ezután botos barátunk mellett a bal szélső ajtón menjünk be. Várjuk meg a patkányinváziót, majd a következő patkánycsapatot átugorva lejtünk tovább egyenesen. Innen től a második pályán szintén a bal szélső ajtó a jó. Továbbhaladva egy éhes krokodil társaságát élvezhetjük. Térjünk vissza az előző terembe, és gyűjtsük meg a fáklyánál a csavargó elől felvett üveget. Ezzel kedveskedjünk a krokodilnak egy jól célzott dobással. A fényes jelenség után látogassuk meg a néhai krokodil lakását.

The basement

Induljunk a ládák között balra (eleinte ez nem lesz könnyű). A verekedés után másszunk fel a létrán és menjünk vissza az előző terembe a mágneskártyáért. Fussunk tovább a galérián, egész annak végéig. Ott lépjünk be a szűk ajtón és szedjük fel a csirkecombot. Térjünk vissza a létrához és másszunk le.

A csillét átugorva térjünk vissza a galéria végéhez, és menjünk ki lent. A „rázós” síneket átugorva vegyük fel a hamburgert. Folytassuk utunkat a ládatornyokig. Ugorjunk először a túlpart felé esőre, onnan a legalsóra, majd tovább a kicsire, és végül a másik oldalra. Máris mehtünk tovább! A laboratóriumon és a vegyszertárakon túli teremben mártsuk a csirkecombot a villogó méregbe. Térjünk vissza a laborba, de most az ajtótól jobbra távozzunk. Nyújtsuk át a mérgezett húst az itt szendergő párdacnak. A következő teremben helyezzük a mágneskártyát a leolvasóba, amire kinyílik az ajtó.

The office

Robogjunk végig a folyosón egészen az első jobbra nyíló ajtóig. Menjünk be és olvassuk le a számítógépről a kódot. Ezt jegegyezzük fel, mert szükségünk lesz rá. Menjünk tovább két szobával, a két irodaajtóhoz. Térjünk be a szembeni irodába. Itt az íróasztalon található egy titkos kapcsoló, ami a mellette lévő ajtót nyitja. Aktivizáljuk a kapcsolót, lépjünk be, másszunk fel a létrán. A fegyverműhely után evickeljünk el a ventilátor előtt. Kapirgáljunk a mellette lévő vasrács alján, mire az felnyílik. Az épület párkányán jobbra indulva egyszélfussunk egészen a létráig. Másszunk fel a folyosón tovább, majd kapaszkodjunk fel a helikopterre.

The mansion

Ugorjunk le a gépről a leghátsó bástyafokra, különben ledobnak. Ugorjunk át az előtte levőre, és onnan a különállóra. Eről hulljunk alá a tetőn lévő vízszintes sávra, és fussunk tovább. A következő pályán esünk be a tetőablakon. Szárguljunk a lépcsőlejáróig, és térjünk be az ott jobbra található szobába. Szedjük fel a kötelet és menjünk le a lépcsőn. A lépcső és a növény között van egy kis nyílás, csúszunk be rajta. Motozzunk a lenti kapcsolótáblán, ami az ajtót nyitja. A mini labirintuson át a kazánházba jutunk. Állítsuk el a gőzcsapot a jobb oldali gombnál, különben megpörkölődünk. Induljunk tovább.

The final battle

Verekedjünk át magunkat a következő terembe. Rántsuk fel az írásjelekkel ellátott függőnyt. Nyomjuk meg a páncélszekrény elejét, majd állítsuk be az Office-ban leolvasott kódot. A szekrény kinyílik, mi pedig vegyük ki belőle a tárgyat. A megjelenő fehér harcos öljük meg a csillag közepén, majd sietve gyűjtsük meg a csillag csúcsein a gyertyákat. A tárgyat visszatesszük a szekrénybe.

Ezek után a gép kifejezi elismerését a gonosz shogunon aratott győzelmünkért.

Tass Csaba



The last ninja II.

BELEÉLÉSTŐL A TÁLALÁSIG

Mindenkivel, aki a számítástechnikát műveli, netán programoz, előfordul, hogy kap egy feladatot, amelyre működő programot kell készítenie.

A programozásnak mindig vannak jó és rossz velejárói, lényege azonban, hogy változatos. Minden feladat más és más, végül is ez adja a munka szépségét. Jó programot csak úgy lehet írni, ha valaki beleéli magát az adott feladatba, látja az értelmét, és gondolatban, hogy a programját, „gyerekét”, „szellemi szüleményét” használni fogják. E felfogás nélkül nem lehet, s nincs is értelme hozzálátni a munkához.

De lássunk neki, kezdjünk el gondolkodni! Sohasem feledjük figyelembe venni a megvalósítás gyakorlati korlátait. Ezen értem a hardverbázist, az anyagi lehetőségeket, az emberi pontatlanságot, a felhasználó kívánságait stb.

Ilyenkor felmerül a kérdés, hogy mi most programozunk vagy épp egy folyamatot szervezünk meg? Erre a kérdésre azt mondhatom, hogy mindkettőt tesszük. Mert a programozás és a szervezés ilyen szinten nem különíthető el egymástól. Ámbar a belegendol-

unk, ez az állításom csak egyoldalú, hiszen a szervező a munkáját a programozó nélkül is el tudja végezni, de ez fordítva nem egészen igaz. Ebből következik, hogy a programozónak egyben szerveznie is kell, s mivel ismeri a feladatot, átlátja, átláthatja a legcél-szerűbb gyakorlati megvalósítást. Viszont, ha az adott feladat oly nagy, hogy több programozónak kell dolgoz-
ni rajta, akkor mindenképpen ajánlatos egy rendszerszervező, aki a feladat legmagasabb szintjén összehangolja az egyes programozók munkáját. De az adott programozóra eső feladat megvalósításába nem szól bele, neki konkrét *bemenő* adatok után konkrét *kimenő* adatok kellene. Ha egy ilyen nagy munkához túl sok szervezőt használunk, akkor a jó rendszer helyett egy túlbonyolított, lassú, nem hatékony valamit kapunk, ez pedig senkinek nem érdeke.

A legfontosabb alappillére a felhasználótól kapott bizonylatok és kimeneti listák terve, erre kell minden be/kiviteli műveletet a későbbiekben felépíteni. Ha a feladat nagysága, milyensége, bonyolultsága megkivánja, akkor mindenképpen folyamatábrát vagy blokk-sémát, illetve számunkra jól értelmez-

hető feladattervet kell készítenie. Ha ezzel megvagyunk, akkor már gondolkodhatunk a gyakorlati megvalósításon. Fontos a be/kiviteli rekordtervek elkészítése; ezt célszerű a felhasználóval közösen megcsinálni a későbbi bonyodalmak elkerülése céljából.

Már csak egy igazán nélkülözhetetlen lépésünk van hátra: a rendszer interaktivitásának megtervezése. Ezen a képernyőterveket és a menüstruktúrákat értem. Lényeges, hogy ne csak jól, megbízhatóan működjön a rendszer, hanem szép is legyen, ami amolyan lelki tükör a felhasználó számára, aki mit sem sejt a program belsejéről. Ebben a fázisban össze kell hangolnunk az esztétikát a praktikussággal, mert egyiknek sem szabad elnyomnia a másikat. Ezt a szakács tudásával és izlésével lehet a legjobban bebizonyítani: ugyebar az ételnek finomnak kell lennie, hogy a kuncsaftok elfogyasszák, de ez nem elég, az étel külleme, esztétikája, tálalása is fontos. Egy jó program ezt mindenképpen figyelembe veszi!

A gyakorlati megvalósítás folyamatában gondolnunk kell a felhasználóra is, aki kezeli a rendszert, tehát gondoskodnunk kell a betanításról és a jó programleírásról is.

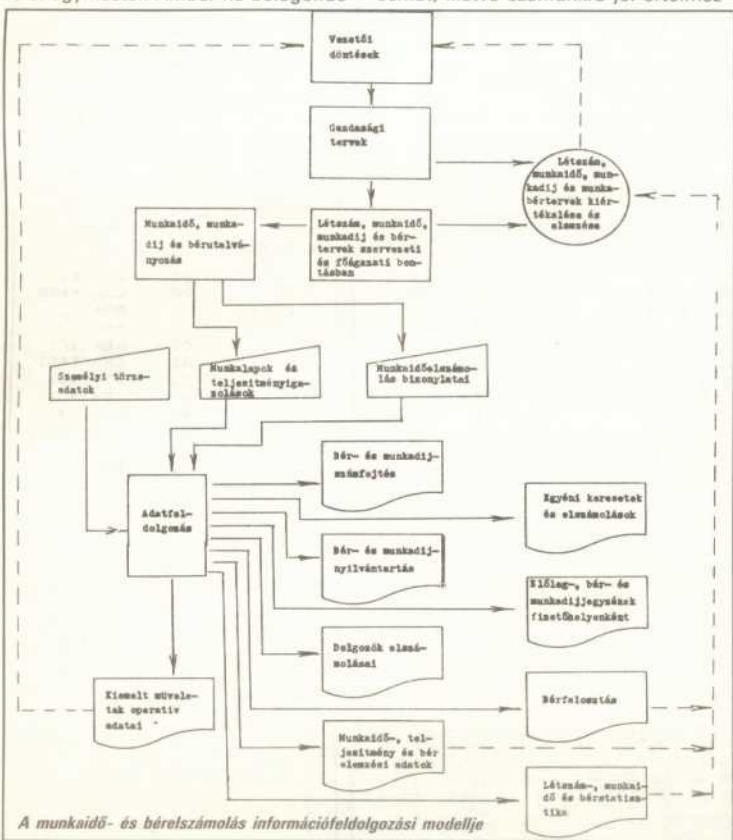
Ahhoz, hogy az itt leírtakról egy kis vizuális fogalmunk legyen, képzéljük el a következő gyakorlati feladatot.

Munkaidő- és bérelszámolás, nyilvántartás

A munkaidő-, munkadíj- és bérelszámolás munkaidőnyelvéssége közismert. Az e területen eddig végzett szervezési munkák tapasztalatai azt igazolják, hogy helyes szervezési megoldásokkal ez a munkaidőnyelvésség jelentősen csökkenthető.

A munkaidőnyelvésség abból ered, hogy a munkaidő-, munkadíj- és bérelszámolás igen sokrétű információfeldolgozást tartalmaz, ezek manuális megoldása többnyire csak külön-külön munkafolyamatokkal biztosítható.

A gépi feldolgozás eszközei már lehetőséget teremtenek arra, hogy az alapp bizonylatok adatait egyidejűleg több szempont szerint feldolgozzák, tehát az e területen igényelt információk kevés vagy együttemű feldolgozásából származnak. Az adatfeldolgozás rendszerének kidolgozásánál alapvető követelmény, hogy a rendszer az alapp bizonylatok együttemű feldolgozásával lehetővé tegye a bérelszámfejtést, a bérgjegyzék elkészítését, a munkaidő- és bérelnyilvántartás vezetését, a létszám-, a munkaidő- és a teljesítményadatok megfelelő részletezésű gyűjtését, a bérelfelosztást, a létszám-, a munkaidő- és a bérelstatistika elkészítését és a munkavállalók különféle tartozásainak nyilvántartását.



A munkaidő- és bérelszámolás információfeldolgozási modellje

Kereséényi István megrendeztetett bennünket azzal, hogy C64-re készült játékok örökélet-kódjait elküldte hozzánk. Saját bevallása szerint kevéssé szorgosztán, egy különböző játékokba ilyen POKE-okat beépít. Úgy gondolta, megosztásán ezeket olvasóinkkal is. Ime hát:

SUICIDE EXPRESS (csak lemezre): POKE 12231,234: POKE 12232,234: POKE 12233,234
STARWARRIOR: POKE 40788,255

GUNFIGHTER (RUN/STOP RESTORE UTÁN):

POKE 49963,255:POKE 49965,0:SYS 50887

CATCH AND KILL: POKE 6397,PEEK(23):SYS 9152

Hatása, hogy a játék elviselhetetlenül lelassul

ANIHILATOR: 6295,11

MRS PAC MAN: 8090,173

COSTA CAPERS: POKE 23843, életek száma

BOMBO: POKE 8887, keret színe (a játék megoldásában ugyan nem könnyít, de érdekes hatásokat érhetünk el vele)

ELITE II: POKE 45229,69:POKE 51211,96 (kalóz úrhajók nem támadnak)

GORF: POKE 7240,234:POKE 7241,234

SMURF: POKE 3221,96 (nem harapós a farkas)

THE GIANT EPIRE: POKE 36211,0:36912,0:36913,0

NEPTUNES DAUGHTER: 7870,80

PLANET DESTROYER: 9926,234 (a keretben nincsenek lövőállások)

PRESENTS III: Irjuk be a listára, ha rekordot döntöttünk: CHEAT 64. Ezek után nem halás egyetlen ellenség sem

FROGGER: POKE 22341,173

WARDERBORDER: Mikor a második lemezoldalról elkezd tölteni, RESET gombbal állítjuk le, majd töltünk be egy C000-nál kezdődő monitorprogramot, írjuk be az M utasítással a 8400-tól kezdődően a következő bájtsorozatot:

M 8400 A6 00 EA EA EA 8D 00 40

M 8408 9D 15 DD EA EA 00 00 00

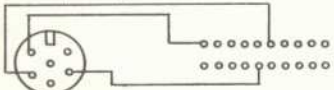
A programot a G 8800 utasítással indíthatjuk monitorból, illetve X utasítás után SYS 34816-tal. Ennek az utasítássorozatnak a hatása: a bolygó közelében a gravitáció elhanyagolhatóvá válik, megkönnyítve az egyébként igen nehézkes manőverezést.



ÖRÖKÉLET

Magyarországban nagyobb számban először a HT-1080Z típusú iskolaszámítógép terjedt el. A hosszú idő alatt, amíg uralta ezt a címet, ha tehetne, sok iskola vásárolt hozzá nyomtatót. Manapság azonban az oktatásban népszerűbbek lettek a Commodore gépek. Ha a C16-hoz vagy C Plus+4-hoz nincs nyomtató, viszont van egy HT és egy hozzá kapcsolt nyomtató, akkor a következő programok segítségével és a két gép összekapcsolásával a C16-ról nyomtathatunk.

Ez természetesen csak példa a felhasználásra, tehát az eljárás alkalmazható — a programok módosításával — adatok továbbítására a C16-ból a HT-ba. Az adatátvitel sorosan működik: a C16 soros bűszát kell a HT digitális be- kimenetével összekapcsolni, az ábra szerint.



A HT programját célszerű SYSTEM paranccsal betölthető formában kazettára venni.

A gépeket lehetőleg kikapcsolat állapotban kapcsoljuk össze. A nyomtatás kezdetekor először a programokat be kell tölteni a gépekbe, majd a HT programját monitorból G7000 paranccsal (vagy SYSTEM/28672) indítsuk el. Miután betöltöttük a C16-ba a nyomtatandó programot, és ha jó a két gép közötti csatlakozás, akkor a SYS 1536 parancs után minden üzenet, amit a C16 ír, a HT nyomtatóján is megjelenik. Tehát ha például a C16-ba betöltött BASIC programot akarjuk kinyomtatni, akkor a LIST paranccsal kilistázzuk, így a lista megjelenik a nyomtató is. Az adatok továbbítása a HT-ba két vezetéken, „handshake” elven lehetséges, ezért a C16 listázásának sebessége a nyomtatóhoz fog igazodni.

A gépeket összekapcsolás háromszálas vezetéken levő soros busz, illetve a HT digitális port csatlakozójának beszerzése sajnos többé-kevésbé gondot jelent. Ennek ellenére Commodore nyomtató hiányában olcsó és egyszerű megoldás a HT-val való összekapcsolás. A programok fejlesztésével akár mindkét gépen létező BA-

SIC utasításokból álló egyszerűbb programok vagy csak adatok is átvihetők a C16-ról a HT-ra.

Gyurkovits Gergely

A HT programja

1	DERKO
2	
3	ORG 7000H
4	LOAD 7000H
5	LD A,7
6	OUT (31),A
7	LD A,128
8	OUT (30),A
9	LD A,15
10	OUT (31),A
11	OUT (30),A
12	LD B,8
13	LD A,14
14	OUT (31),A
15	IN A,(31)
16	RRA
17	JR C,HT-2
18	LD A,15
19	OUT (31),A
20	XOR A
21	OUT (30),A
22	LD A,13
23	DEC A
24	JR NZ,*-1
25	LD A,14
26	OUT (31),A
27	IN A,(31)
28	RRA
29	RL C
30	LD A,15
31	OUT (31),A
32	OUT (30),A
33	LD A,13
34	DEC A
35	JR NZ,*-1
36	DDNZ HT
37	LD A,C
38	CALL 003BH
39	CALL 0033H
40	JP 7000H
41	END

A C16 programja

.0600	A9 C8	LDA	#C8
.0602	85 01	STA	#01
.0604	A9 0F	LDA	#0F
.0606	A2 06	LDX	#06
.0608	8D 24 03	STA	#0324
.060B	8E 25 03	STX	#0325
.060E	00	BRK	
.060F	78	SEI	
.0610	86 D0	LDX	\$D0
.0612	A2 08	LDX	##08
.0614	48	PHA	
.0615	48	PHA	
.0616	A5 01	LDA	#01
.0618	09 01	ORA	#01
.061A	85 01	STA	#01
.061C	A5 01	LDA	#01
.061E	29 40	AND	##40
.0620	D0 FA	BNE	#061C
.0622	68	PLA	
.0623	0A	ASL	
.0624	48	PHA	
.0625	08	PHP	
.0626	A5 01	LDA	#01
.0628	4A	LSR	
.0629	28	PLP	
.062A	2A	ROR	
.062B	49 01	EOR	##01
.062D	85 01	STA	#01
.062F	A5 01	LDA	#01
.0631	29 40	AND	##40
.0633	F0 FA	BEG	#062F
.0635	A5 01	LDA	#01
.0637	29 FE	AND	##FE
.0639	85 01	STA	#01
.063B	CA	DEX	
.063C	D0 DB	BNE	#0616
.063E	68	PLA	
.063F	68	PLA	
.0640	A6 D0	LDX	\$D0
.0642	58	CLI	
.0643	4C 4B EC	JMP	##E4B

A számítógépek motorja III.

Erőforrás-kezelés

Alapelvek

Amikor még a számítógépes rendszerekben a hardver ábra dominált, erőforráson a rendszer fontosabb hardverkomponenseit értették: a központi egységet, a memóriát, a B/K csatornákat és a különböző perifériákat. Az idők folyamán ez a szemlélet megváltozott, az erőforrások köre a szoftver erőforrásokkal, azaz programokkal és adatokkal bővült. A szoftver erőforrások — mivel ezek élőmunka-tartalma magas és egyre növekvő — értéke emelkedik, a hardver erőforrásoké pedig élőmunka-tartalmukkal együtt csökken. Gondoljunk például egy kiterjedt bankhálózathoz tárolt adatok értékére; elvesztésük jóformán helyrehozhatatlan kárt okozna.

Általában elmondható, hogy az operációs rendszerek egyike fontos funkciója a teljes rendszer erőforrásainak kezelése. Ez azt jelenti, hogy az operációs rendszer gondoskodik az erőforrások elosztásáról, védett használatáról és az elosztás során kialakuló versenyhelyzetek kezeléséről.

Az erőforrás kezelésekor az operációs rendszer maga is létrehoz később elosztandó erőforrásokat. Ilyenek például a virtuális tárkezelésnél a lapok, a B/K során használt pufferek, vagy akár az adatállományok.

Az erőforrás-kezelés két alapvető célja a számítógéprendszer működésének valamiképpen gazdaságossá tétele és a holtponthelyzetek kivédése. A két cél annyiban összefügg, hogy ha lemondanánk a gazdaságos üzemről, történetesen az erőforrások osztott felhasználásáról, a holtpont kérdése fel sem merülne. A fejezetben az erőforrással való gazdálkodásnak csupán magasabb szintű stratégiaiál foglalkozunk, a fontosabb konkrét erőforrások — processzor, memória, állományok, programok — ütemezésének külön fejezeteket szentelünk.

Gazdaságosság

A számítógép használati értékét elsősorban az a haszon határozza meg, amit a gép használata közvetlenül vagy közvetve hajt. Ilyen lehet a termelési költség csökkentése, az adatok tárolása, a felhasználók termelékenységének növekedése és az információk minőségileg értékebb és jobban ütemezett szolgáltatása. A bevételt természetesen terhelik a hardverköltségek, az üzemeltetés és az alkalmazási rendszerek fejlesztési vagy beszerzési költségei.

A gazdaságos üzem valamilyen költségfüggvény értékének minimalizálását jelenti. A probléma általában ott van, hogy a költségfüggvény nem univerzális, hanem függ a számítógépes rendszer felhasználásának körülményeitől. Az operációs rendszerek általában két — sajnos egymásnak ellentmondó — költségfüggvénnyel szoktak dolgozni. Az egyik az erőforrások minél teljesebb kihasználtságára törekszik, a másik a rendszer választesztjét igyekszik optimalizálni. Az általános stratégia kívül megfogalmazhatók — nem ellentmondásmentes — konkrét célok is:

- az átlagos átfutási idő minimalizálása,
- a lassú válaszok számának minimalizálása,
- a hardver kihasználtságának maximalizálása,
- maximális kihasználtság adott válaszidő-korlát esetén.

Elvileg az a jó — és vannak is ilyen operációs rendszerek —, ha egy nagy rendszerrel az üzemeltető befolyásolhatja az erőforrás-kezelési stratégiát.

Az elmúlt húsz év alatt a hardver árával együtt a gazdaságosság megítélése is változott. A tendencia az, hogy a maximális kihasználtság háttérbe szorul a válaszkészséggel szemben. A másodjegyű rendszereknél ez mindig is így volt, de újabban általánossá vált az időosztásos rendszerekben való fejlesztési tevékenység is, ami szintén a válaszkészségre helyezi a hangsúlyt. A személyi számítógépek elterjedése ugyancsak ebbe az irányba hatott.

A lefoglalás munkafázisa

A klasszikus nagygépes feldolgozásban egy ütemező az indítható feladatok közül olyat próbál választani, amelyhez az erőforrások — memória, adatállományok és perifériák — adottak voltak. Ilyenkor a két esemény, az ütemezés és az erőforrás lekötése összekeveredett: a rendszer az F1 feladathoz például lefoglalta a memóriát (mert ezt meg tudta tenni), majd üzenetet küldött az operátornak, hogy tegye fel azt a mágnesszalagot, amelyen az A1 állomány van. Ezután az ütemező esetleg az F2 feladathoz is foglalt memóriát és egy lemezfajlit, majd üzent egy másik mágnesszalagért. E tevékenységek eredményeként még egyik feladat sem indult el, de már le van kötve a memória és két szalagos egység. A feladat csak akkor indul, ha az ütemező minden erőforrásigényét „összegyűjtötte”. Ezt a módszert nevezhetnénk statikus lefoglalásnak is, hiszen az erőforrásokat a feladat indítása előtt lekötik. Ennek az a nagy hátránya, hogy az erőforrások akkor is függő — más feladat számára nem használható — állapotban lehetnek, amikor tulajdonképpen még senki nem használja őket. A statikus rendszerben egy feladat meghatározhatatlan ideig várakozhat az indítható munkák egymásutánjában, ameddig sorra kerül. Ugyanakkor, ha egy feladat végrehajtása elindult, erőforrás hiánya miatt nem szakad meg.

A statikus lefoglalást természetes módon szorította ki a dinamikus módszer, amelyben az indítható feladatot minden előzetes erőforrásigény kielégítése nélkül választjuk ki. A szükséges erőforrásokat a feladatot maga kérheti futása alatt az operációs rendszertől. Az ütemező csak azt teszi lehetővé, hogy a feladat részt vehet az erőforrásokról folyó versenyben. A felhasználónak persze előny a gyors indítás, ami természetesen még nem garantálja a gyors befejezést. A rendszer számára az éppen nem használt erőforrások szabadja tétele és újra hasznosíthatósága a fontos, ami azonban függ a programozótól is. Ilyen stratégiával nagymértékben növekedhet a rendszer átbocsátóképessége, ha minden folyamat csak addig tart, ameddig arra okvetlenül szüksége van (például a programozó nem feledkezik meg a már feldolgozott állományok azonnali lezárásáról). Dinamikus stratégiát követnek a modernebb kötegel, valamint az időosztásos rendszerek, ahol a statikus módszer szöbe sem jöhetne.

Holtpont

A holtpont tulajdonképpen az erőforrásokkal gazdálkodó, azok megosztott használatát lehe-

tővé tevő algoritmusok mellékhatása, melynek kezelése egy fejlett operációs rendszerből sem hiányozhat. A folyamatok egy halmaza akkor kerül holtponthelyzetbe, ha mindegyikük egy olyan esemény bekövetkezésére vár, amit csak a halmaz valamelyik folyamata tud kiváltani. Az erőforrásokkal kapcsolatban az esemény értelmelemzerük azok lekötése és elengedése lehet. A dinamikus lefoglalási stratégiának megfelelően egy folyamat a következő sorrendben kerül egy erőforrással kapcsolatba:

— Igénylés. Ha az igény nem teljesíthető azonnal — például az erőforrást egy másik folyamat használja —, az igénylő várakozásra kényszerül.

- Használat.
- Felszabadítás.

Az igénylés és a felszabadítás mindenképpen az operációs rendszer privilegizált rutinjainak hívását jelenti, a használat elvileg a felhasználó hatáskörében is megoldható, de általában az alsó szinten itt is a rendszerutrinok működnek.

A holtponthelyzet kezelése érdekében kidolgoztak egy — a holtpont bekövetkezéséhez szükséges — feltételrendszer.

E szerint egy rendszer csak akkor kerülhet holtponthelyzetbe, ha a következő négy feltétel egyidejűleg teljesül:

- Kölcsönös kizárás. Legalább egy erőforrás nem osztható módon foglalt, azaz egyszerre csak egy folyamat használhatja. Ha egy másik folyamat is igényli, várnia kell az erőforrás felszabadulásáig.
- Várás közben lekött. Van olyan folyamat, amely lekött legalább egy erőforrást, miközben várakozik más folyamatok által használt erőforrás(ok)ra.
- Rablás nincs. Az erőforrások egy folyamatot erőszakkal nem vehetők át. A felszabadított a folyamat mindig önkéntesen végzi.
- Ciklikus várakozás. Létezik a folyamatok egy olyan $\{p_0, p_1, \dots, p_n\}$ halmaza, hogy p_0 egy p_1 által lefoglalt, p_1 egy p_2 által lefoglalt, ..., p_n egy p_0 által lefoglalt erőforrásra várakozik.

A feltételek nem teljesen függetlenek, a „ciklikus várakozás” például tartalmazza a „várás közben lekött” feltételt.

A holtpont alapvetően kétféleképpen kezelhető: bizonyos szabályok betartásával megelőzhető, vagy bekövetkezését felismerjük, majd a rendszert fellesztjük. Mint várható, a megelőzés az „olcsóbb”, mi csak ezzel foglalkozunk.

A holtpont megelőzése

Arról kell gondoskodni, hogy a négy szükséges feltétel közül legalább egy ne teljesüljön. Ennek érdekében mindegyiket közelebbről is meg kell vizsgálni.

A kölcsönös kizárás feltételével nem sokra lehet jutni. Ez olyan eszközökre vonatkozik, amelyeket nem használhat két folyamat párhuzamosan (pl. egy sornyomató). A párhuzamosan használható erőforrásokat viszont nem célszerű kölcsönösen kizáró módon kezelni, ezért ezek nem játszanak szerepet a holtpont kialakulásában. Ha például egy csak olvasható állományt tekintünk, azt akárhány folyamat használhatja egyidejűleg, tehát arra senkinek nem kell várnia.

Programozási fogások és



melléfogások

A sorozat eddigi három részében a gazdaságtalan algoritmusok kiválasztására mutattam példákat. Legutóbb **A BASIC nyelvű programozás abcéje** című könyvből idéztem a szerzők által javasolt algoritmust és programot a legnagyobb közös osztó kiszámítására, azután egy sokkal gyorsabb eljárást mutattam be ugyanezre. Akkor elmulasztottam megírni azt, hogy az említett könyv tankönyv, mégpedig **Dusza Árpád—Varga Antal: A BASIC nyelv c. tankönyv változatlan kiadása**. Erről a tankönyvről olvastam a „ régi” Számítástechnika 1986. júliusi számának 15. oldalán egy könyvismertetést, melynek idézem néhány mondatát:

„Dusza Árpád és Varga Antal könyve igaz-vérgig hagymányos tankönyv, mégpedig kiemelkedően a legjobb eddig a magyar piacon. (...) Ha ezt a könyvet kapta volna minden középiskola a HT géphez, akkor a magyar iskolaszámítógép-program nem félkarú óriás lenne a közepesen fejlett világ törpéi között.”

Ezek után képzeljük el, hogy milyen programok jelennek meg a **nem kiemelkedően legjobb** tankönyvekben.

Az itt következő példák olyan könyvekből választottam ki, melyek nem tankönyvek ugyan, de nem titkolt céljuk a programozási fogások oktatási célú bemutatása. A bemutatásra kerülő hibák nem súlyosak, veszélyük gyakoriságukban van, s főleg abban, hogy a programok szerzői műveiket követendő példaként állítják a tapasztalatlan olvasók elé, így terjesztve a rossz programozási stílust. Nem utolsósorban terjengőségükkel felesleges munkával terhelik azokat, akik kipróbálás céljából begépelik programjaikat.

Téglás Erzsébet Példák gazdasági jellegű feladatok megoldására BASIC nyelven című könyvét 1987-ben (!) adta ki az OMIKK. Ime az első feladat a könyv 7. oldaláról: „Ijunk programot, amely egy hétjegyű telefonszámról megállapítja, hogy budapesti-e!” A megoldás rövid ismertetéséből kiderül, hogy a budapesti telefonszámot arról ismerhetjük fel, hogy 1-es számjeggyel kezdődik. A 8. oldalon található meg az *1/a listán* látható ötsoros program. Most tekintsünk el attól az apróságtól, hogy a ciklusból csak a STOP és a RESTORE billentyűk egyidejű lenyomásával lehet kilépni, ezt a könyv szerzője is csak átmeneti megoldásnak szánta. A második sorban a THEN és a GOTO együttes használata felesleges, az 50-es sor END utasítása szintén. Ami nagyon csúnya, az a második sorban lévő GOTO utasítás, amely egy másik GOTO-ra ugrat. Érthetetlen, miért nem lehet rögtön a 10-es sorra ugrani? Próbáljuk meg!

További egyszerűsítésre ad módot, ha az IF utasítás feltételét megfordítjuk. A módosított program az *1/b listán* látható.

A következő három példa **Könözi László—Gál István—Vasvári József Csupa játék C16-ra** című könyvéből van, mely a Műszaki Könyvkiadónál jelent meg

```
10 INPUT "TELEFONSZAM"; SZ$
20 IF LEFT$(SZ$,1)<>"1" THEN GOTO 40
30 PRINT "BUDAPESTI"
40 GOTO 10
50 END
```

1/a lista

```
10 INPUT "TELEFONSZAM"; SZ$
20 IF LEFT$(SZ$,1)="1" THEN PRINT "BUDAP
ESTI"
40 GOTO 10
```

1/b lista

```
...
280 IF A$="I" THEN 300
290 GOTO 310
300 GOTO 320
310 X9=0 : Y9=0 : GOTO 340
320 ...
```

2/a lista

```
...
280 IF A$="I" THEN 320
310 X9=0 : Y9=0 : GOTO 340
320 ...
```

2/b lista

```
...
280 IF A$<>"I" THEN X9=0 : Y9=0 : GOTO 3
40
320 ...
```

2/c lista

```
...
590 IF X1<1 THEN GOTO 640
600 IF X1>25 THEN GOTO 640
610 IF Y1<1 THEN GOTO 640
620 IF Y1<40 THEN GOTO 640
630 GOTO 340
640 ...
```

3/a lista

1987-ben. A programrészleteket eredeti környezetükből kiemelve, de lényegükön nem változtattam. Ezek is jól illeszkednek a GOTO megfontolatlanság használatát.

A könyv 55. oldalán egy torpedójátékot mutatnak be. Idézek a bevezető szövegből: „A játékot dr. Kocsis András (...) könyve alapján egy szakkörön elemeztük, és úgy találtuk, hogy kezdő programozók számára több tanulságos szerkezeti elemet tartalmaz, tehát C16-os gépen is ajánlatos a programot bemutatni.” Egy tanulságos szerkezeti elemet látunk a 2/a listán. Javítását a 2/b listán mutatom be, további egyszerűsítése — melyben az eredeti négy programorból egy lesz — a 2/c listán látható.

A 3/a listán ugyanennek a programnak egy másik részlete látható. Ebből azt tanulhatjuk meg, hogy hogyan lehet egy (legfeljebb két) programsort ötre növelni a hatékonyság növelése nélkül. Nem tartom valószínűnek, hogy a szerzők a hosszú programsortól ijedtek meg, hiszen a könyv más helyén 88 karakternél hosszabb — több utasítást tartalmazó — programsort is láttam, amely utasítás rövidítés nélkül be sem írható. A helyes megoldás a 3/b listán van. A 3/c listán látható változat még jobb, de ismerni kell hozzá a logikai műveletek átalakításának szabályait.

Utolsó példánk a könyv 46. oldalán kezdődő löverseny-programból való. A 4/a listán bemutatott szerkezet a programban összesen háromszor fordul elő, különböző adatokkal. Az idézett részben egy programhiba is van, melyet csak a feladat ismeretében fedezhetünk fel: a 370-es sorban 1001 helyett 1000 lenne helyes. Az END utasítás itt feleslegesebb, mint az első példában, mert ott legalább programzáró funkciót lehetne belemagyarázni, de itt még azt sem. Az egyszerűsítés módja a 4/b listán látható.

Barna László

```
590 IF X1<1 OR X1>25 OR Y1<1 OR Y1>40 TH
EN 640
630 GOTO 340
640 ...
```

3/b lista

```
590 IF X1>0 AND X1<26 AND Y1>0 AND Y1<41
THEN 340
640 ...
```

3/c lista

```
...
370 IF PENZ>1001 THEN 400
380 GOTO 420
390 END
400 ...
```

4/a lista

```
...
370 IF PENZ<1001 THEN 420
400 ...
```

4/b lista

Legyen-e sorellenőrző kód?

A Magazin ez évi 1. számának 39. oldalán S. E. részletesen kifejti, hogy a lapban közölt programlisták miért nem tartalmaznak ellenőrző számokat. Bár abban egyetértek vele, hogy ezek közlése nehézségekbe ütközik, úgy gondolom, hogy a cikkben leírt érveinek nagy része nem helytálló.

Szükségtelen, hogy a kódellenőrző program a lapnak minden számában megjelenjen, és az sem szükségszerű, hogy minden géptípushoz legyen ilyen program. A hazai példaként felhozott Mikrovilág programozási rovatában 1988-ban öt különböző géptípusra közölték programokat, de kódellenőrző programot csak C64-hez és C Plus/4-hez adtak. (A C16-ost itt nem tekintem külön géptípusnak, mivel programozás szempontjából nem tér el a C Plus/4-től.) Az említett Commodore gépek mindegyikére az év folyamán több mint húsz számban jelent meg egy vagy több program, de a kódellenőrző programok összesen négy-négy alkalommal jelentek meg.

Külföldi példára is hivatkozhatok. Az NSZK-ban megjelenő „kevésgépes” 64-es magazin rendszeresen közöl programokat C Plus/4-re is, de ellenőrző kód nélkül. A C64-esre készült Checksummer 64 sem jelenik meg minden számban, bár a lap terjedelme ezt megengedné. A V3 jelű változat az 1985/8. számban közölték, az utána következő legalább öt számban egyszer sem, legfeljebb annyit, hogy megcímzett és a postai szállításhoz elegendő bélyeggel ellátott válaszkártyák ellenében bárkinek megküldik a listáját

a gépi kódú programok begépelését segítő MSE program listájával együtt.

S. E. állításával ellentétben a BASIC programok listáiban az ellenőrző kód kevés helyet foglal el, nem növeli számottevően a terjedelmét, és ez is könnyen kompenzálható.

A példaként felhozott 1988/10. számban közölt BASIC programokhoz — a Tandy Co-Cora írt egyetlen egysoros (sőt „egysorosok”) program kivételével — jócskán elkelne az ellenőrző kód, ha másért nem, legalább azért, hogy az olvasó biztos legyen abban, nem ő gépelt be rosszul a szöveget, hanem valóban a közölt program működik hibásan. Arra a C Plus/4-re készült programra gondolok itt, melyet fáradságos munkával beírtam a gépbe, aztán ugyanannyi — ha nem több — munkát fordítottam az esetleges gépelési hibák keresésére. Végül kiderült, hogy a program nemcsak feleslegesen hosszú, de hibásan is működik.

Valós gond, hogy szükség lenne kódellenőrző programokra, ezeket el kellene készíteni. Valószínűnek tartom, hogy valamely külföldi lapból való átvétel nem úszható meg anyagi vagy jogi következmények nélkül, bár tudok néhány ilyen irányú kísérletről. Zsom Béla a Formázott listázás című, kétrészes írásában (Mikroszámítógép Magazin, 1987/9—10. szám) javasolja a Compute's Gazette PROOFREADER nevű kódellenőrző programjának átvételét, közölte a magyar szövegű programlistát is. Tudomásom szerint azóta semmi sem történt e témával kapcsolatban.

S. E. cikkében olvastam Gregorits József javaslatát, hogy vegyük át a Mikrovilág kódellenőrző programját. Ez még kevésbé járható út, mint az előbbi. Egyrészt azért, mert a Mikrovilág ragaszkodik a megjelent írásokkal kapcsolatos jogaihoz, másrészt azért, mert az említett program — tudomásom szerint — idegen szellemi termék jogtalan kisajtítása, olyasmí, amit szépirodalmi berkekben plágiumnak neveznek. Akkor inkább maradjunk Zsom Béla javaslatánál, mert a PROOFREADER négy különböző Commodore géptípus kódellenőrzését oldja meg egy-maga.

Másik probléma — amit S. E. is említ — a listázás. A kódellenőrző számot alkalmazó lapoknak a programokat általában mágneses adathordozón — lemezen vagy magnókazettán — kell beküldeni, és azokat központi egységesen listázzák. Ez nálunk sajnos megoldhatatlan, de arra kellene törekedni, hogy a szerkesztőség a külső munkatársaktól is csak jól használható programlistákat fogadjon el, s ha az ellenőrző kódokat bevezetnénk, lehetőleg a kód feltüntetésével együtt. A Commodore gépekre Zsom Béla említett cikkében erre is láthatunk megoldást.

Nagyon örülünk, ha a Mikroszámítógép Magazin is esatlakozna az ellenőrző kódot alkalmazó lapok táborához, annak ellenére, hogy sokgépese lap. Úgy gondolom, sok bosszúságtól és felesleges fáradságtól mentesülnek az olvasók.

B. L.

Rekurzió átalakítása

iterációvá

A programozásemélet irodalma elég sokat foglalkozik rekurzív algoritmusok nem rekurzívvá való átírásával. Legtöbbször azonban megállnak annál, hogy a jobb, illetve bal rekurziót ciklussá írják át. Jobb rekurzióról beszélünk, ha a rekurzív hívás az eljárás végén található, és bal rekurzióról, ha a hívás az eljárásba belépés után következik. Átírás szempontjából hasonló a helyzet, ha a rekurzív hívás az eljárás közepén található. A jobb rekurzió átírása különösen egyszerű; sima ciklussá szervezhető. A bal rekurzió már egy fokkal bonyolultabb: egy verem szervezése és kezelő utasításainak megírása után két különálló ciklussá esik szét a feladat. Az első ciklusban sorra letesszük (push) a megőrizni kívánt elemeket, majd a másik ciklusban szabadítjuk fel (pop) az elemeket. A szerzők általában bemutatják, hogy milyen feladatoknak van olyan tulajdonságuk, hogy ily módon átirhatók. Az utóbbi időben is találkozhatott a magyar olvasó ilyen tárgyú cikkel, illetve könyvvel.

Ebben a cikkben azonban a fentieknél általánosabb rekurzív algoritmusok ciklussá átírását mutatjuk be. Célunk, hogy a gyakorlatban előforduló, de nem ilyen „szabályosan” viselkedő algoritmusokkal foglalkozzunk. Nehéz megfogalmazni az ilyen algoritmusok általános szerkezetét. Röviden: olyan algoritmusokra gondolunk, amelyekben a rekurzív hívások száma nem egy, hanem többszörös. A jobb megértés kedvéért egy konkrét algoritmusból kiindulva szeretnénk az általános megfogalmazási és átírási szabályig eljutni.

Nézzük példának a bináris fa preorder bejárásának rekurzív algoritmusát!

A bináris fa egy-egy szögpontjáról a következő adatokat tároljuk:

```
! BAL ( CIM ) ! szögpon ti elem ! JOBB ( CIM ) ! , vagyis
```

```
-----
baloldali részfa      jobboldali részfa
címek                  címek
```

A terminálisok BAL(CIM)-e, illetve JOBB(CIM)-e 0, /

Az egyszerű rekurzív algoritmus a következő:

```
PREORD eljárás (CIM)
MŰVELET (CIM)
Ha BAL (CIM) > 0 akkor
    PREORD (BAL(CIM))
    PREORD (JOBB(CIM))
Elágazás vége
PREORD vége.
```

Egy szögpontból 0 vagy 2 élt tételezünk fel.

Természetesen most is adott az a gondolat, hogy rekurzió ciklussá szervezését egy veremmel oldjuk meg. A preorder bejárás lényegéből kitűnik, hogy itt nincs meg az a szabályosság a veremlépésekben, mint az említett cikkek feladataiban. Ez a bejárás a „felülről lefelé”, „balról jobbra” jelszavakkal jellemezhető. Vagyis először egy szögpontból kiindulva, annak legmélyebb szintű bal oldali terminálisáig haladunk (PRE-ORD)-szinten: első hívások egymásutánja, VEREM-szinten push műveletek egymásutánja, majd az ezen éle kapcsolódó részfák bejárása következik, sorrendben alulról felfelé. (PREORD-szinten a két rekurzív eljárás hívása, illetve az azokból való visszaté-

rések váltakozása; VEREM-szinten szintén a push és pop műveletek előre láthatatlan sorozata.)

A továbbiakban egységés magyar nyelvű elnevezéseket használunk: a push eljárást LETESZ eljárásnak, a pop eljárást FELSZABADÍT eljárásnak nevezzük.

Algoritmusunk váza a következő:

```
PREORD eljárás (CIM):
.
.
VEREMLEPÉS ciklus
Elágazás
    LETESZUNK esetén
        LETESZ (CIM)
        MŰVELET (CIM)
    FELSZABADITUNK esetén
        FELSZABADIT
Elágazás vége
VEREMLEPÉS ciklus vége
PREORD eljárás vége.
```

Látjuk, hogy egy teljesen általános veremkezelésen alapuló eljárás szerkezetét adtuk meg. Ábrázoljuk szemléletesen Jackson-féle diagrammal:



A tevékenységek:

- 1 LETESZ (CIM)
- 2 MŰVELET (CIM)
- 3 FELSZABADÍT

A doboz jobb sarkában található X jel az iterációt, a 0 jel a szelekiót jelenti.

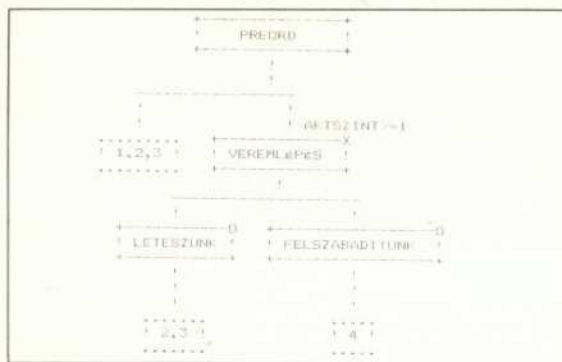
A következő lépésben meg kellene határozni a konstrukciós műveletek feltételeit. Első kérdés: meddig hajtsuk végre a veremlépés ciklust? Hogy determinisztikus legyen a szerkezet, alkalmazzuk az előleolvasás analógiájaként az első veremlépést, azaz a verem inicializálása (fájlnyitás) után egy előrehozott

LETESZ (CIM)
MŰVELET (CIM)

utasításlépést — egyszerűs előleolvasást — alkalmazunk. A verem aktuális szintjét mutatja egy AKTSZINT változó (az egyszerűség kedvéért globális változóként tekintünk): alkalmas a ciklus terminálására (van_még_rekord, illetve nem fájl_vég analógiájára).

Ekkor a pontosított programozási szerkezet az alábbi:

bán: Nézzük meg, hogyan valósulnak meg a fentiek pseudokódok:



A tevékenységek:

- 1 INICIALIZÁL
- 2 LETESZ (CIM)
- 3 MŰVELET (CIM)
- 4 FELSZABADÍT

A következő lépés a szelekció kiértékelése. A verem egy szintjén nem elég a változó(ka)t tárolni — esetünkben a szögpont címét —, hanem információra van szükségünk arról, hogy az azon a szinten lehetséges rekurzív hívásoknak megfelelő veremműveletek közül melyiket kell aktualizálnunk. A verem a következőképpen szervezhető jól:

>Aktív tárolása rekurzív hívásállapotok kezelésére szolgálnak rész

	11	1	1
	11	1	1
	11	1	1
	11	1	1
	11	1	1
	11	1	1
AKTÍVSZINT	11	1	1
	11	1	1
	11	1	1
	11	1	1
	11	1	1
	11	1	1

CIMVÁLTOZÓ BALFA JOBBFA

A javasolt szimbolikus konstansok:

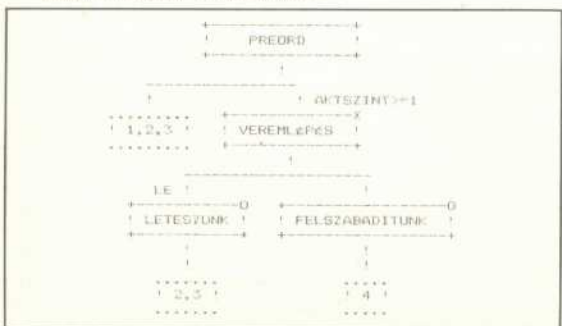
BEJÁRATLAN = HAMIS
BEJÁRT = IGAZ

A verem értelmezéséhez a következőkre van szükségünk:

— a BEJÁRATLAN és BEJÁRT szimbolikus konstansokkal hamis, illetve igaz értékeket adhatunk a BALFA és JOBBFA elemeknek,

— egy logikai függvénnyel meg kell állapítanunk, hogy lefelé vagy felfelé irányuló veremlépés következik; az aktuális szinten bejártunk már minden részfelt. Az eljárás neve legyen LE!

Ekkor a diagram az alábbi lesz:



```

PREORD eljárás(CIM):
    INICIALIZÁL eljárás
    LETESZ eljárás(CIM)
    MŰVELET eljárás(CIM)
    VEREMLÉPÉS eljárás(CIM)
    ha LE(CIM) akkor
        LETESZ eljárás(CIM)
        MŰVELET eljárás(CIM)
    különben
        FELSZABADÍT eljárás
    VEREMLÉPÉS eljárás vége
PREORD eljárás vége

INICIALIZÁL eljárás:
    SZINT := 0
    AKTÍVSZINT := VEREMLÉPÉS
    VEREMLEVELEK[AKTÍVSZINT, JOBBFA] := BEJÁRATLAN
    VEREMLEVELEK[AKTÍVSZINT, BALFA] := IGAZ
    LE(CIM) vége
    INICIALIZÁL eljárás vége

LETESZ eljárás(CIM):
    VEREMLEVELEK[AKTÍVSZINT, BALFA] := BEJÁRATLAN
    AKTÍVSZINT := AKTÍVSZINT + 1
LETESZ eljárás vége

FELSZABADÍT eljárás:
    VEREMLEVELEK[AKTÍVSZINT, BALFA] := BEJÁRATLAN
    VEREMLEVELEK[AKTÍVSZINT, JOBBFA] := BEJÁRATLAN
    AKTÍVSZINT := AKTÍVSZINT - 1
FELSZABADÍT vége

LE logikai eljárás(CIM):
    ha BAL(CIM) = 0 akkor
        LE = HAMIS
    különben
        ha VEREMLEVELEK[AKTÍVSZINT, BALFA] = BEJÁRATLAN akkor
            LE = IGAZ
        VEREMLEVELEK[AKTÍVSZINT, JOBBFA] := BEJÁRATLAN
        CIM := BAL(CIM)
    különben
        ha VEREMLEVELEK[AKTÍVSZINT, JOBBFA] = BEJÁRATLAN akkor
            LE = IGAZ
        VEREMLEVELEK[AKTÍVSZINT, JOBBFA] := BEJÁRATLAN
        CIM := JOBBFA(CIM)
    különben
        LE = HAMIS
    Eljárás vége
LE eljárás vége
    
```

Általánosan elmondhatjuk, hogy azok a rekurzív algoritmusok, amelyekben valamilyen predikátum teljesülésének függvényében többszörös meghívásra kerül sor, a fent említett módon átirítható meg rekurzívra:

— a fenti vezérlő modul egy az egyben, a veremkezelő eljárások szinte egy az egyben alkalmazhatók, csak

— a verem „állapotjelző” részét kell a feladathoz igazítottan definiálni, valamint

— feladatonként a LE logikai eljárást meg kell írni.

Látható, hogy a javasolt veremszerkezet lényegében kétdimenziós. Az első dimenziója a szokásos felhasználást jelenti a változók tárolására, a második dimenzió pedig az egy szinten való rekurzív hívások állapotait fejezi ki.

Belátható, bár itt nem alkalmunk bizonyítani, hogy a kétdimenziós veremszerkezettel és a definiált algoritmusokkal olyan rekurzív sémák is átiríthatók ciklussá, amelyek a tetszőleges számú programváltozót megengedő folyamatábrára számájába nem transzformálhatók.

Írásuk egyébként nem csupán elméleti eszmefuttatás. Valóság feladatok megoldásaként alakult ki. Nevezetesen egy bonyolult rendszerprogramozási feladatot kellett megoldani olyan nyelvi környezetben (FORTRAN), amelyben nem volt rekurzív.

A Jackson-féle programtervezési módszert alkalmazva kényszerült rá a fejlesztő a szerkezetek ilyen tiszta és általános kialakítására és a szerkezet által sugallt veremszerkezet megfogalmazására.

Bánné dr. Varga Gabriella

Rekurzió átalakítása iterációvá

A BASIC programok tárolása a memóriában

Az Enterprise is, mint más számítógépek, a BASIC programokat az értelmezés meggyorsítása érdekében átalakítva, kódolva tárolja a memóriájában.

Az Enterprise BASIC memóriája a nullás lap szegmensén a 4827 (= 12DBh) címen kezdődik. Mivel ez a szegmens (248 – OF8h) normális esetben mindig a nullás lapon van, ezért a BASIC programok ezen a címen mindig elérhetők. Az ennél nagyobb címek ezen a szegmensén csak a BASIC editornak vannak fenntartva, de természetesen kellő óvatossággal itt is elhelyezhetünk gépi kódú programokat. A programok a memóriában tárolva is megtartják soros felépítésüket, és a sorok sorszáma szerinti sorrendben állnak. Egy memóriában tárolt BASIC-sor felépítése a következő:

- 1. bájt:** a sor memóriabeli hossza bájtokban. E bájt címéhez hozzáadva ezt az értéket, megkapjuk a következő sor első bájtjának a címét.
- 2–3. bájt:** a sor sorszáma bináris alakban — tehát elől az alacsonyabb helyiértékű (LSB), utána a magasabb helyiértékű bájt (MSB) — van.
- 4. bájt:** blokkmélység. Ennek a bájtjának az értéke mondja meg a listázó rutinnak, hogy az áttekinthető, tördelt listázási kép érdekében ezt a sort milyen mélyen kell megjeleníteni. Alapértelme 0, ekkor a sor közvetlenül a sorszáma mögé íródik ki. Ha 1 az értéke, akkor két plusz szököz szűrődik a sorszáma mögé, ha 2, akkor 4 szököz stb.

5-n. bájt: a BASIC utasítások kódolva.

n+1. bájt: 00, azaz a sorvégjel (ERW).

Ezután közvetlenül a következő sor adatai jönnek, ugyanilyen struktúrában. Hogyha a soron következő BASIC-sor első bájtjának, azaz a hosszúságbájtjának az értéke nulla, ez a program végét jelzi. Egyébként a NEW utasítás sem csinál mást, mint az első programsor hosszúságbájtját törli. Az BASIC-értelmező tehát azt hiszi, hogy nincs a tárbán érvényes program. Ennek ismeretében a NEW utasítással megmondhatóan kívülől programunkat „restaurálni” tudjuk.

Ezek után talán érthető a kézikönyvnek az a kitétele, hogy egy BASIC program sor maximálisan 250 karakterből állhat: ugyanis a sor tőrből képehez hozzájön még plusz öt bájt. Így kaphatjuk meg az egy bájt-on maximálisan ábrázolható 255-ös értéket. A kézikönyv szintén írja, hogy a programsorszámok 1 és 9999 között lehetnek. Ez így is van, ha a BASIC-szerkesztővel dolgozunk. Am utólag például POKE utasítással felülírhatjuk a sorszámokat 0 és 9999-nél nagyobb értékre. Ekkor is értelmes, futtatható programot kapunk, csak a sorszámok helyett értelmetlen karakterek írődnek ki, és a programsort nem lehet editálni, módosítani. Ezzel elérhetjük azt, hogy avatation kezek ne tudjanak belekörtárodni programjainkba, vagy mi ne tudjuk véletlenül kitorolni a sort.

A sor érdemi része nem ASCII kódok formájában tárolódik, hanem ún. BASIC kódokkal. Ezek kódolhatnak egyrészt karaktereket — például kettőspontot —, utalhatnak arra, hogy utánuk milyen típusú adat következik — történetesen sztring —, illetve lehetnek tokenek. A tokenek BASIC kulcsszavak helyett álló kódokat jelentenek. A mellékelt táblázatban megtalálhatók a BASIC kódok és a tokenek.

A különleges kódok rövidítéseinek magyarázata:

ERW (End of Row): a sorvégjel. Ez jelenti a BASIC-értelmező számára azt, hogy az adott sor befejeződött.

FP (Floating Point): e kód után álló 6 bájt az értelmező lebegőpontos számként próbálja értelmezni. A 6 bájt felépítése a következő: BY4 BY3 BY2 BY1 BY0 EXP, ahol a BY szimbólumú bájtok adják a mantisszát, az EXP pedig az exponenst. Az exponens 7. bite az előjelbit: ha az 1 értékű, a szám negatív lesz. A maradék bitek értékeiből 64-et (40h) levonva megkapjuk az exponens tényleges értékét. A mantisszát BCD számok alkotják. Itt négy biten ábrázolódik egy tizes számrendszerbeli számjegy. Az öt bájtból a legelső (BY4) adja a legalacsonyabb helyiértékű számjegyeket, a legutolsó (BY0) pedig a legmagasabbakat. Az egy bájtban ábrázolt két számjegy közül az elsőnek van nagyobb helyi értéke. A tizedespontra a legnagyobb értékű bájt (BY0) két digitje közé értendő. Eszerint a 35428 a következőképpen ábrázolódik:

35428 = 3.5428 × 10⁴
 dec.: 0, 0, 8, 66, 53, 68
 hexa.: 00 00 08 42 35 44

INT (Integer): e kód után 2 bájt-on ábrázolt egész szám következik. Az Enterprise csak a -9999...9999 intervallumba eső

számokat kódolja így, de megérti az ilyen alakban megadott nagyobb számokat is. Emiatt ezt az alakot kis egész alaknak is hívják. A szám maga rendszeren gépi ábrázolása, tehát a bináris értelmű bájtok LSB MSB sorrendben követik egymást.

RWN (Row Number): e kód teljesen az INT mintájára felépülő egész száma utal, amint az értelmező BASIC sorszámmal fog fel. Tehát ilyen adat áll a GOTO, GOSUB stb. utasítások után.

STR (String): ezt a kódot egy karakterfüzér, sztring követi. Az ábrázolás ASCII kódokkal, a határoló idézőjelek nélkül lehetséges. A STR karaktert egy hosszúbájt, majd a sztring karakterei követik.

NUV (Numeric Variable): az ezt követő karakteresorozat egy BASIC változó azonosítója, illetve egy olyan tokenné az nem íródó BASIC kulcsszó neve, amely nem „S” karakterre végződik. Bővebbet a következő kódnál.

STV (String Variable): e kódra következő karakteresorozat egy sztringváltozó azonosítója, illetve egy olyan tokenné az nem íródó BASIC kulcsszó neve, amely „S” karakterre végződik. Mindkét kódból 32 darab van. Ennek az az oka, hogy ebben a kódban vonták össze az azonosítók hosszát is. A kódok 0–4. bitjei adják meg a név hosszát. Emiatt lehet egy változó azonosítója maximum 31 karakteres.

TKN (ToKeN): a következő bájt egy BASIC utasítás tokenje. Több szóból álló utasításoknál csak az első szó tokenizálódik, a többi változóazonosító formájában kódolódik. Használóképpen nem írődnek át tokenné a függvények, opciók nevei sem.

A programlistában lévő szóközök nem tárolódnak, azokat az értelmező listázóprogramja surzja be értelemszerűen. Egyébként a fent említett változásokkal leszámítva a BASIC program térbeli szintaxisa meg egyezik az editorba való begépelésorok eloiratkaik.

Az elmondottak megvilágításához álljon itt egy mintaprogram és annak memóriabeli képe decimális, valamint hexadecimális számokban.

```
100 OPTION ANGLE DEGREES
110 LET ALPHA = 23: LET MAX = 35428
120 LET NAMES$ = „ENTERPRISE”
130 FOR I = 0 TO 100 STEP 2
140 LET P = PEEK(MAX + I): LET PS$ = STR$(P)
150 PRINT#102, AT ALPHA,3:P$; ” ”;
160 NEXT I
170 GOTO 120
180 END
```

A decimális memórialista soronként:

21,100,0,0,96,50,57,65,78,71,76,69,39,68,69,71,82,69,93,0
32,110,0,0,96,40,37,65,76,80,72,85,19,194,23,0,16,96,40,35,77,85,88,19,194,0,0,120,69,53,68,0
26,120,0,0,96,40,69,78,65,77,89,36,19,128,10,68,78,84,69,82,80,82,73,83,69,0
27,120,0,0,96,31,23,75,18,194,0,0,34,34,79,194,100,0,36,83,84,86,80,194,20,0
40,140,0,1,96,40,23,80,19,36,80,65,89,75,8,39,77,65,88,11,23,73,9,16,96,40,86,80,36,19,65,83,94,32,36,8,33,80,9,0
34,150,0,1,96,56,3,194,102,0,12,34,65,84,37,65,76,80,72,65,17,194,0,0,16,66,80,36,17,128,1,72,17,0
9,160,0,0,96,47,35,73,0
10,170,0,0,96,23,162,120,0,0
7,180,0,0,96,33,0
0,0,0

A hexadecimális lista:

15 64 00 00 60 32 26 41 45 47 42 45 27 44 45 47 52 45 45 53 00
20 62 00 00 60 28 29 41 40 50 48 41 13 02 17 00 10 60 20 23 47
1A 78 00 00 50 28 45 4E 41 40 45 24 13 80 0A 45 4E 54 52 50
52 49 53 45 00
1B 82 00 00 60 1F 21 49 13 02 00 00 22 54 4F C2 64 00 23 53 54
45 50 22 00 00
23 8C 00 01 60 28 21 50 13 24 50 45 45 4B 08 23 40 45 58 01 21
49 05 10 60 21 42 50 24 13 44 53 54 52 2F 09 21 50
05 00
22 9C 00 01 60 28 02 02 65 00 02 41 54 25 41 40 50 4B 41 9C
C2 03 00 10 42 50 24 11 60 01 20 11 00
09 A0 00 00 60 2F 21 4F 00
0A A4 00 00 60 21 47 78 00 00
07 B4 00 00 60 17 00
00 00 00

A BASIC program után közvetlenül a változó területe kezdődik, vagyis ha a szerkesztőt megkerülve bővíti programunkat, megtörténhet, hogy az értelmező nem ismeri a BASIC programterületnek új végcímét, a változókkal felülírja azt. Ezért ilyenkor célszerű néhány felesleges REM sorral helyet biztosítani az illegális programbővítésnek.

A BASIC KÓDOK ÉS TOKENEK TÁBLÁZATA

dec	hex	kód	token	dec	hex	kód	token
0	00	<ERW>	ALLOCATE	53	35	<NUV>	PLOT
1	01		ASK	54	36	<NUV>	POKE
2	02	<ERW>	AUTO	55	37	<NUV>	SPOKE
3	03	#	CALL	56	38	<NUV>	PRINT
4	04	\$	CAPTURE	57	39	<NUV>	PROGRAM
5	05	%	CASE	58	3A	<NUV>	RANDOMIZE
6	06	&	CAUSE	59	3B	<NUV>	READ
7	07		CLEAR	60	3C	<NUV>	REDIRECT
8	08	(CLOSE	61	3D	<NUV>	REM
9	09)	CODE	62	3E	<NUV>	RENUMBER
10	0A	*	CONTINUE	63	3F	<NUV>	RESTORE
11	0B	+	COPY	64	40	<STV>	RETRY
12	0C	.	DATA	65	41	<STV>	RETURN
13	0D	-	DEF	66	42	<STV>	RUN
14	0E	-	DEF	67	43	<STV>	SAVE
15	0F	/	DELETE	68	44	<STV>	SELECT
16	10		DIM	69	45	<STV>	SET
17	11	<	DISPLAY	70	46	<STV>	SOUND
18	12	<	DO	71	47	<STV>	START
19	13	>	CHAIN	72	48	<STV>	STOP
20	14	>	EDIT	73	49	<STV>	INFO
21	15	<>	ELSE	74	4A	<STV>	STRING
22	16	<=	ELSE IF	75	4B	<STV>	TEXT
23	17	>=	END	76	4C	<STV>	TOGGLE
24	18	Å	END DEF	77	4D	<STV>	TRACE
25	19	Ø	END HANDLER	78	4E	<STV>	TYPE
26	1A	Û	END IF	79	4F	<STV>	VERIFY
27	1B		END SELECT	80	50	<STV>	WHEN
28	1C		END WHEN	81	51	<STV>	!
29	1D		ENVELOPE	82	52	<STV>	LLIST
30	1E		EXIT	83	53	<STV>	LPRINT
31	1F		FOR	84	54	<STV>	EXT
32	20	<NUV>	GOSUB	85	55	<STV>	GET
33	21	<NUV>	GOTO	86	56	<STV>	FLUSH
34	22	<NUV>	GRAPHICS	87	57	<STV>	LOOK
35	23	<NUV>	HANDLER	88	58	<STV>	PING
36	24	<NUV>	IMAGE	89	59	<STV>	DATE
37	25	<NUV>	IF	90	5A	<STV>	TIME
38	26	<NUV>	IF	91	5B	<STV>	WAIT
39	27	<NUV>	INPUT	92	5C	<STV>	ON
40	28	<NUV>	LET	93	5D	<STV>	
41	29	<NUV>	LINE	94	5E	<STV>	
42	2A	<NUV>	LIST	95	5F	<STV>	
43	2B	<NUV>	LOAD	96	60	<TKN>	
44	2C	<NUV>	LOOP	:	:		
45	2D	<NUV>	MERGE	128	80	<STR>	
46	2E	<NUV>	NEW	:	:		
47	2F	<NUV>	NEXT	162	A2	<RWN>	
48	30	<NUV>	NUMERIC	:	:		
49	31	<NUV>	OPEN	194	C2	<INT>	
50	32	<NUV>	OPTION	:	:		
51	33	<NUV>	OK	198	C6	<FP>	
52	34	<NUV>	OUT				

Megjegyzés a táblázathoz:

Mint látható, a táblázatban két DEF, illetve IF token szerepel. A -tal jelölt tokenek egyetlen BASIC sorban elhelyezkedő utasításra, míg a -tal jelölt utasításblokkot kezdő parancsra vonatkoznak.

Az ON szimbólum az ON...GOTO utasítástokenje, sohasem az opciókapcsoló ON szócské.

A LINE INPUT utasításban csak a LINE tokenizálódik, az INPUT nem; ha azonban az INPUT önmagában alkot utasítást, akkor átíródik tokené.

A Mikroszámítógép Magazin 1988/8. számában Dósa Istvánnak a 26–27. oldalon megjelent cikkét (Memória, grafika, billentyűzet) szeretném megkérdésíteni. A billentyűzetcsoporthoz leírt 2. táblázatnak része még két billentyű is ezeknek a kódjai. Az x = 9-es csoporthoz tartozó I, O, P billentyűk után tehát még két sort kell illeszteni:

```
x=9
I      254
O      251
P      239
U      247
+      223
```

Ezenkívül a táblázat szemléletesebbé is tehető: kérdéses ugyanis, hogy milyen értéket kapunk akkor, ha az I és a P billentyűket egyszerre tartjuk lenyomva. Ez a probléma akkor fordul elő a leggyakrabban, ha a botkormány állását kívánjuk megállapítani, és szükségünk van a ferde állás leolvasására is. A billentyű ábrázolásának szokásosabb módja a billentyűmátrix megadása. Ilyenkor a 181-es portra kiírandó értékek a táblázat sorait, míg a beolvasáskor kapott bájttitjei a mátrix oszlopait adják. A billentyűmátrix német tasztatúrájú gépeknél a következő:

a PORT—181-re írandó értékek

0:	a portról kiolvasott bájttitjei:						
	0.	1.	2.	3.	4.	5.	6.
0:	N	>	B	C	V	X	Y
1:	H	<LOCK>	G	D	F	S	A
2:	U	Q	Z	R	T	E	W
3:	F	1	6	4	5	3	2
4:	<F4>	<F8>	<F3>	<F6>	<F5>	<F7>	<F2>
5:	8	9	6	0			
6:	J	K	0	L	Å		*
7:	<STOP>	<DWN>	<RGT>	<UP>	<HOLD>	<LFT>	<ENT>
8:	M				<RTSH>	<SP>	<INS>
9:	I		O	U	P		+

Ha a billentyű éppen le van nyomva, az 0t reprezentáló bit 0 értékű, ugyanakkor ha nincs lenyomva, a bit értéke 1. Eképpen könnyen kiszámítható, hogy ha a botkormány jobbra—fel állású és előzőleg a portra 7-et írunk, a leolvasott bájttitje értéke 243. Ugyanígy I és P együttes nyomásakor, ha 9 lett a portra kiírva, a kapott érték 238 lesz.

A jelölésekről: RTSH jelöli a jobb oldali, LTSH a bal oldali SHIFT-billentyűt. A botkormány állása: DWN — le, RGT — jobbra, UP — fel, LFT — balra. Az SP jelöli a betűköz-billentyűt, az ENT az ENTER-t, az ERA az ERASE-t, a HOLD felel meg a német gépeken a PAUSE billentyűnek.

Vigyáznunk kell arra — különösen leltöltött megszakításakor —, hogy a 181-es port felső 3 bitje más értelmű. A 6. és a 7. bitnek állítják be a távvezérlő kimeneteket, az 5. bit pedig a magnetofon ki-bekapcsolását vezérli.

Racsó Tamás

Fedezzük fel együtt!

A beépített függvények számát tekintve nem lehetünk panaszosra. Az ábécé szerinti felsorolásuk a Felhasználói kézikönyv 153. oldalán kezdődik. Mi most más rendezőelvet követünk. A függvényekből csoportokat alakítunk ki az egyszerűbb eligazodás kedvéért. A csoportosításunkon kívül természetesen más szempontok alapján is lehet osztályozni a függvényeket: ismerem — nem ismerem, már alkalmaztam — még nem alkalmaztam stb. Ezek a megkülönböztetések is segítségünkre lehetnek a tanulásban.

Az Enterprise belső függvényeit a következőképpen csoportosíthatjuk:

A színfüggvények: BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE és az RGB.

Miután ezeket már ismerjük, a soknak tűnő függvények száma máris csökkent. Az RGB függvényről még lesz szó a 27. programban.

A matematikai függvények. Ezekből van a legtöbb, ezért további csoportokat hozunk létre. Egyparaméteresek: nevéket a 28. program DATA listájában találjuk meg. Kétparaméteresek: ezeket a 31. és 32. listán mutatjuk be. Átalakítást végzők: DEG, RAD és BIN. Tömbfüggvények: LBOUND, SIZE, UBOUND. Alkalmazásukra a 28. programban, a 475–478-as sorokban találunk példát. Beolvasó és kiíratási függvények (beolvasó: IN, JOY, kiíratási: TAB). A számítógép állapotát jelző függvények:

Függvényeink függvényében

INF, PEEK, SPEKK, FREE, EXTTYPE, EXLINE, FBS. A gépi szubrutin hívása: USR.

A színekről már volt szó, ennél fogva nem ismeretlenek előttünk a színek megadására szolgáló függvények. Arról, hogy ezek a kulcsszavak tényleg függvények, meggyőződhetünk, ha kiadjuk például a PRINT BLUE parancsot. A függvény értéke a két szín kódja lesz.

A 27. listán levő programmal azt a 256 színt mutatjuk be, amivel a számítógépen dolgozhatunk. Az RGB függvény paramétereivel az alapszínek — piros, zöld, kék — arányait adhatjuk meg. A megjelenítés helyének kiszámításához a MOD és az INT függvényeket alkalmazzuk.

30. lista

```
100 REM --- 27. program ---
110 LET L=0
120 GRAPHICS HIRES 256
130 FOR R=125 TO 1 STEP .125
140 FOR B=125 TO 1 STEP .125
150 FOR G=25 TO 1 STEP .25
160 PRINT "szinkodi";RGB(R,G,B)
170 PRINT " R:";R, " G:";G, " B:";B
180
190 SET INK RGB(R,G,B)
190 LET J=32*MOD(L,32)+1;LET I=64+
INT(L/32)
200 PLOT 120+J,600-I;120+J,600-I
-32
210 LET L=L+1
220 GET Q#
230 IF Q#="" THEN 220
240 NEXT
250 NEXT
260 NEXT
```

27. lista

```
100 REM --- 28. program ---
110 GRAPHICS HIRES 4
120 PLOT 640,719;640,0
130 PLOT 0,360;1279,360
140 LET E=100
150 FOR I=0 TO 1279 STEP 4
160 LET X=(I-640)/E
170 LET Y=SIN(X)
180 LET J=360+E*Y
190 IF J<0 OR J>719 THEN 210
200 PLOT I,J
210 NEXT
```

28. lista

29. lista

```
100 REM --- 29. program ---
110 GRAPHICS HIRES 4
120 PLOT 640,719;640,0
130 PLOT 0,360;1279,360
140 LET E=100;LET V=0
150 SET INK 3
160 WHEN EXCEPTION USE HIBA
170 FOR I=0 TO 1279 STEP 4
180 LET X=(I-640)/E
190 LET Y=COT(X)
200 LET J=360+E*Y
210 PLOT I,J
220 NEXT
230 END WHEN
240 END
250 REM --- hibakezelo ---
260 HANDLER HIBA
270 PRINT EXTTYPE
280 CONTINUE
290 END HANDLER
```

```
100 REM --- 30. program ---
105 REM --- függvények ábrázolása ---
110 TEXT
120 DIM Q(30)
130 NUMERIC K,U,V
140 GOSUB 430
150 LET E=100
160 LET A=1;LET B=1279
170 IF V=2 OR V=3 THEN LET A=640-E;LET
B=640+E
180 IF V=14 OR V=15 OR V=16 THEN LET A
=641
```

31. lista

```
100 REM --- 31. program ---
110 TEXT
120 INPUT AT 3,2,PROMPT "a,b (b)0:";A
,B
130 IF B=0 THEN
140 PRINT "0-val nem lehet osztani!"
150 END
160 END IF
170 PRINT AT 5,3;"MOD(A/B)=" MOD(A/B);
180 IF B<0 THEN 200
190 PRINT TAB(20) "A-B=INT(A/B)=" A-B*
INT(A/B)
200 PRINT AT 7,3;"REM(A/B)=" REM(A/B);
210 PRINT TAB(20) "A-B=IP(A/B)=" A-B*
IP(A/B)
220 PRINT AT 9,3;"CEIL(A)=" CEIL(A);
230 PRINT TAB(20) "-INT(-A)="-INT(-A)
240 PRINT AT 11,3;"A/B=" A/B
250 PRINT AT 13,3;"FP(A/B)=" FP(A/B);
260 PRINT AT 15,3;"A/B-IP(A/B)=" A/B-IP
(A/B)
270 RANDOMIZE ;LET N=RND(6)-3
280 REM 1st rnd(6rnd)-3
290 PRINT AT 17,3;"ROUND(A/B, N)="
ROUND(A/B,N)
300 PRINT AT 19,3;"INT(A/B*10^N)-N *10^
5)/10^N " = " INT(A/B*10^N)-N)/10^N
```

32. lista

```
100 REM --- 32. program ---
110 TEXT
120 INPUT AT 3,2,PROMPT "a,b";A,B
130 LET K=MIN(A,B);LET L=MAX(A,B)
140 PRINT AT 5,4;"a,b5 egész számai:"
CHR$(15)
150 FOR I=CEIL(K) TO INT(L)
160 PRINT I;
170 NEXT
180 PRINT
```

```
190 IF V=20 THEN LET A=640
200 IF V=17 THEN LET A=0
210 GRAPHICS HIRES 4
220 GOSUB 410
230 GOSUB 270
240 END
250
260 REM --- ábrázolás ---
270 FOR I=A TO B STEP 4
280 LET X=(I-640)/E
290 GOSUB 730
300 LET J=360+E*Y
310 LET T=5
320 IF J<0 THEN LET T=0;LET J=0
330 IF J>719 THEN LET T=B;LET J=719
340 SET INK T
350 IF (V>10 AND V<14) OR V=17 OR V=
5 THEN
360 PLOT I,J
370 ELSE
380 PLOT I,J;
390 END IF
400 NEXT
410 RETURN
420
430 REM --- kiválasztás ---
440 LET N=1
450 DO
460 READ Q#(N)
470 LET N=N+1
475 IF UBOUND(Q#)+1<N THEN
476 PRINT "A DIM-ben a felső index
et növelni kell!"
477 END
478 END IF
480 LOOP UNTIL Q#(N)=""
490 LET N=N-2;LET V=1
500 PRINT AT 20,4;"keresés:joy / kiv
választás-space"
510 PRINT AT 12,15;" ;VICHRS(161);PRI
NT AT 12,20;" ;Q#(V);" ( )"
520 LET W=JDY(0)
530 IF W=15 THEN RETURN
540 IF W<0 AND W>4 THEN 520
550 IF W=0 THEN LET V=W+1
560 IF W=4 THEN LET V=W+1
570 IF W=1 THEN LET V=1
580 IF V=N+1 THEN LET V=1
590 GOTO 510
600
610 REM --- koordinátarendszer ---
620 PLOT 620,710,:PRINT #10;" Y"
630 PLOT 1250,370,:PRINT #10;">"
640 PLOT 620,324,:PRINT #10;"X"
650 PLOT 640,710;640,0
660 PLOT 0,360;1279,360
670 PLOT 640+E,360;640+E,352
680 PLOT 632+E,320,:PRINT #10;"1"
690 SET INK 2
700 PLOT 30,700,:PRINT #10;Q#(V);" (X)
710 RETURN
720
730 REM --- függvények ---
740 ON V GOSUB 760,770,780,790,800,810
,820,830,840,850,860,870,880,890,900,910
,920,930,940,950,960,970
750 RETURN
760 LET Y=ABS(X):RETURN
770 LET Y=ACOS(X):RETURN
780 LET Y=ASIN(X):RETURN
790 LET Y=ATN(X):RETURN
800 LET Y=CEIL(X):RETURN
810 LET Y=COS(X):RETURN
820 LET Y=COSH(X):RETURN
830 LET Y=COT(X):RETURN
840 LET Y=DCOS(X):RETURN
850 LET Y=EXP(X):RETURN
860 LET Y=FP(X):RETURN
870 LET Y=INT(X):RETURN
880 LET Y=IP(X):RETURN
890 LET Y=LOG(X):RETURN
900 LET Y=LOG(X):RETURN
910 LET Y=LOG2(X):RETURN
920 LET Y=SGN(X):RETURN
930 LET Y=SIN(X):RETURN
940 LET Y=SINH(X):RETURN
950 LET Y=SQR(X):RETURN
960 LET Y=TAN(X):RETURN
970 LET Y=TANH(X):RETURN
980 DATA abs,acos,asin,atan,ceil,cos,co
sh,cot,csc,exp,fp,int,ip,log,log10,log2
,sgn,sin,sinh,sqr,tan,tanh,*
```

Az egyváltozós matematikai függvények vizsgálatához, megismeréséhez három programot készítettünk. A 28. listán látható program elég rövid. Tekinthető „egy-két sorosnak” is. Sajnos, az Enterprise-tulajdonosok az „egy-két soros programok” írásában nem rughatnak labdába, mert a gép csak áttekinthető programok írását segíti elő és teszi lehetővé. Lehet, hogy az utasítások száma jobban jellemzi a program készítőjének ötletességét?

Rövid programunk azonban jó néhány dolgra hibát jelez. Most egy nagyon durva megoldást adunk, ami arra lesz jó, hogy megismerkedjünk a hibák kezelésével.

A 29. listáról beírható programba beépítettünk egy hibafigyelést és az azt lekezelő HANDLER-blokkot. A WHEN-blokkba azt a programrészt tettük, ahol várjuk a hiba megjelenését. A hiba bekövetkezésekor a végrehajtás a HANDLER-blokkban folytatódik. Itt most csak a hiba kódját írjuk ki az EXTYPE függvényvel. A CONTINUE hatására a hibát jelző sort követő soron folytatódik a program. Egy másik, a RETRY utasítással arra a sorra lehet visszatérni, amelyekben a hiba bekövetkezett.

A különböző függvények ábrázolásakor — ha a hibakódokat figyeljük — az is kiderül, hogy mi az, amit vizsgálnunk kell. Mire figyeljünk?

— Ismernünk kell a függvények értelmezési tartományát.
— Azokat a pontokat, amelyek nem esnek a képernyőre, ne próbáljuk meg kirajzolni!

Nézzük ezek után a matematikai függvények legismertebb és legnagyobb csoportjának bemutatását a 30. program segítségével, amelyek több szubrutinból áll. Vizsgáljuk meg mindegyiket a végrehajtásuk sorrendjében!

Kiválasztás szubrutin. Egyszerű „menü”, ahol először feltöltjük a QS vektort a függvény nevével. Itt látunk példát az egyik tömbfüggvény alkalmazására.

A JOY függvényrel a botkormány helyzetét kérdezzük le, aminek a mozgásával kikereshetjük az ábrázolandó függvényt. A választás a SPACE billentyű lenyomásával lehetséges. Ez lesz a beépített, azaz a 0-ás botkormány „tűz” billentyűje.

A DATA-ba újabb függvényeket is írhatunk a * karakter elé. A DIM utasítás harminc névnek ad helyet (a 0. indexet nem használjuk). Ha új függvényeket írunk be, meg kell határoznunk még az értelmezési tartományt és az ábrázolás módját is. Ez utóbbi azt jelenti, hogy pontokkal vagy vonalakkal rajzoljunk-e.

Koordináta-rendszer szubrutin. Bemelő paraméterként az E-t, azaz az egységnek választott képernyőpontok számát és a függvény sorszáma, a V-t kell megadni.

Ábrázolás szubrutin. Mielőtt belépünk a szubrutinba, határozzuk meg az értelmezési tartományt, azaz A és B értékét.

A szakadásos függvények közül néhánynak az ábrázolását úgy oldottuk meg, hogy egy kis ügyeskedéssel kikerültük azokban a pontokban a függvény értékének kiszámítását, ahol nincs értelme. Ezért lesz A=1. Az SGN függvényben az SGN(0) értékének ábrázolása miatt A=0. X>0 esetén A=641, X>=0-nál A=640. A 11, 12, 13, 17-es és az 5-ös sorszámú függvények ábráját pontokból alakítottuk ki, a többinél összekötöttük a pontokat (350–380-as sorok).

Függvények szubrutin. Ezzel az ON GOSUB utasítást is bemutatjuk. Újabb függvény beírásakor az utasítássor sorszámtól ide is be kell írni. A függvények közül az INT, IP, CEIL és az FP függvényekre szeretném felhívni a figyelmet. Alkalmazásukra bemutatunk néhány példát a 31. lista programjával. Van ugyan néhány kétparaméteres függvény, amelyekkel ezek az átalakítások, számítások egyszerűbben megoldhatók, de úgy gondolom, érdemes összehasonlítani a megoldásokat.

Eddig még az RND függvényvel nem foglalkoztunk. Nem is soroltuk be egyik csoportba sem. A két alak miatt nem döntöttem: RND(6) a hatnál kisebb nem negatív egész számok közül választ véletlenszerűen (270-es sor). Ugyanez történik a 280-as sorban is — a REM-ben mutatom be — az RND-vel. Az RND 0 és 1 közé eső véletlenszámtól állt elő. Hármat azért vonunk ki, hogy negatív számokat is kapjunk.

Gyakorlásul próbáljuk megoldani a következő feladatot: írassuk ki az [A:B] egész számait! Másféppen fogalmazva: írassuk ki az A-nál nem kisebb és B-nél nem nagyobb egész számokat, ha A<B teljesedik! A megoldást hasonlítsuk össze a 32. listán látható programmal!

Dusza Árpád

Mi a manó?

Magyarország – Budapest. Sok panaszos levelet kapunk vidéki Enterprise-tulajdonosoktól, hogy nem jutnak hozzá helybéli áruházakban programokhoz, szakirodalomhoz és kiegészítőkhöz. Sajnos saját országjáró körutainkon ugyanezt tapasztaltuk. Nincs jó hírnök sem, mivel részben a vidéki áruházak felkészületlensége (a szakértelem hiánya), részben az elmúlt évben kialakult Centrum – Novotrade viszály miatt sem, illetve csak nagyon mérsékeltlen rendelnek számítástechnikai eszközöket. A kialakult helyzettel ugyan egyáltalán nem értünk egyet, azonban csak azt tanácsolhatjuk az Enterprise-osoknak, hogyha biztosra akarnak menni, a kijelölt márkáruházakat keressék fel. Ezek Budapestén vannak! Emiatt a vidékieknek sajnos utazniuk kell. A szakosodott üzletek a következők: az Út-törő, az Otthon, a Kispesti és a Novotrade 2C Áruház. Nagyon nem örülünk annak, hogy e remek és olcsó gép körül újabb vihar dül.

*

LORIGRAPH-kiterjesztés. Minden bizonnyal sok Enterprise-tulajdonosnak megvan a LORIGRAPH rajzoló-program. Ennek egyik jó tulajdonsága, hogy egyszerűen átirható vele a gépünk karakterkészlete, és az szalagra is menthető. A következő rövid program a szalagra mentett karakterkészlet más, saját fejlesztésű programjainkban való felhasználását teszi lehetővé.

```

10 ALLOCATE 50
20 CODE KAR = HEX$ ("18,06")
30 CODE TAP = HEX$ ("05,54,41,50,45,3A")
40 CODE = HEX$ ("3E,6A,11")
50 CODE = WORD$ (TAP)
60 CODE = HEX$ ("F7,01,3E,6A,01,80,B4,F7,
06,3E,6A,3E,6A,F7,03,C9")
500 CALL USR(KAR,0)
    
```

A programot Fendrik László küldte be.

Monogram. Írjunk az ST FLAG rendszerváltóba (1Ah=26) 42-t. Érdekes dolog történik. Az állapotosorban a ROM programok íróinak monogramjai jelennek meg.



Geatsch Günterné rajza

Hardver

A sorozat alap gondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a programozónak

rendelkeznie kell alapfokú áramkört hardverismerettel is. Megegyezik ez, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretre.

MIKROPROCESSZOROS RENDSZEREK TERVEZÉSE ÉS FEJLESZTÉSE

A cikksorozat eddigi részeiben összefoglaltuk azokat az alapvető hardverismereteket, amelyekre alapozva hozzáfoghatunk kisebb mikroprocesszoros rendszerek fejlesztéséhez.

A fejlesztést lépésről lépésre haladva mutatjuk be, többek között azért, hogy eloszlassuk a mikroprocesszoros rendszerek fejlesztésével kapcsolatos misztikus légkört, és egyszerű, jól alkalmazható gyakorlati tanácsokat adjunk a rendszereket készítőknak. Az ábrán összefoglaltuk a rendszerfejlesztés lépéseit.

A mikroprocesszoros rendszerek fejlesztésével kapcsolatos alapvető nehézség abban van, hogy egy ilyen feladat megoldása kettős tevékenységet igényel:

- meg kell tervezni és valósítani a működő áramkört, a hardvert,
- meg kell írni a működtető és az adott feladatot a kialakított hardverrel megvalósító programot, a szoftvert.

Mivel ez a két tevékenység szorosan kapcsolódik egymáshoz, ezért nélkülözhetetlen, hogy a fejlesztő bővében legyen mindkét témakör alapvető ismereteinek. Az ilyen rendszerek tervezői és fejlesztői már nem „szoftveres” vagy „hardveres” szakemberek, hanem mindkettőt — bár az egyiket általában jobban — fejlesztők.

A cikksorozat eddigi részeiben sem törekedtünk minden kérdés pontos, részletes tárgyalására, és a továbbiakban is azt a gyakorlati utat kívánjuk bemutatni, hogyan lesz egy feladat meghatározásából az azt megvalósító mikroprocesszoros rendszer.

Teljes általánosságban minden ilyen rendszer úgy tekinthető, hogy a rendszer bemenetére jutó jelekből valami-

lyen algoritmus — módszer — alapján előállítja a kimenő jeleket.

A továbbiakban először összefoglaljuk a rendszerfejlesztéssel kapcsolatos elméleti alapismereteket, majd egy konkrét, jól használható mikroprocesszoros rendszert is bemutatunk.

A mikroprocesszort tartalmazó rendszerek létrehozásakor a következő lépéseket kell elvégeznünk: a feladat megfogalmazását, az áramkörti tervezést, a működtető program megtervezését, a kódolást, a hibakeresést, a programvizsgálatot (tesztelést) és a dokumentálást.

Egy működő berendezés megvalósításánál mindegyiknek nagy jelentősége van. A kódolás — ami a program megírását jelenti — csupán egy a többi között, és nem is mindig a leglényegesebb. Foglaljuk össze az egyes lépésekkel kapcsolatos legfontosabb ismereteket!

A FELADAT MEGFOGALMAZÁSA

A mikroprocesszorra megoldandó feladatok szükségessé teszik számos tény és körülmény pontos megszabását. Feltétlenül tisztázandó az alapvető kérdés, hogy valóban szükséges-e mikroprocesszoros rendszer alkalmazása.

A bemenetek definiálása

Először a rendszer bemeneteit kell definiálni. Fel kell tehát sorolni, hogy a feladat megoldása milyen bemeneteket igényel, és jellemezni is kell azokat.

— Milyen a bemenő jelek formája, időbeli viselkedése?

— Mi jelzi a bemenő jel aktív állapotát?

— Milyen hosszú ideig aktív a bemenő jel?

— Milyen gyakran — gyorsan — változik a bemenő jel, képes-e a program a bemenő jel változását követni?

— A bemenő jel kapcsolatban van-e más bemenő vagy kimenő jelekkel?

A kimenetek definiálása

Ehhez hasonló kérdéseket kell megválaszolni a rendszer kimeneteinek definiálásakor is.

A feldolgozó rész definiálása

Ez az a programrész, ami a bemenő jelekből és adatokból előállítja a kimenő jeleket és adatokat:

— Mi az alapvető algoritmus, ami a bemenetek alapján megadja a kimeneti jeleket?

— Milyen időkorlátok vannak? Milyen gyorsan kell a feldolgozást végre hajtani?

— Milyen memóriaterület-korlátozások vannak? Van-e valami korlát a programot tartalmazó memóriarészre vagy az adatokat tartalmazó memóriarészre vonatkozólag?

— Milyen szabványos — esetleg már meglévő — programrészleteket, szubrutinokat használhatunk fel?

— Milyen speciális esetek vannak, a program hogyan kezeli ezeket?

— Az eredményeknek milyen pontosnak kell lenniük?

— Hogyan tudja a feldolgozás során fellépő hibákat a program kezelni?

Hibakezelés

Nagyon fontos a program működése közben fellépő hibák felderítése és

felelő módon való kezelés. A hibakezeléssel kapcsolatban a következő szempontokat kell figyelembe venni:

- Milyen hibák lehetségesek?
- Mely hibák a legvalószínűbbek? (Ha emberi beavatkozás is van, az emberi hibák a leggyakoribbak.)
- Milyen hibák maradhatnak rejtve egy ideig a rendszerben?
- A rendszer hogyan tudja a hibát a legkevesebb következménnyel elhárítani?

Az emberi tényező

Sok mikroprocesszor alapú rendszer emberi közreműködéssel dolgozik. Az ilyen rendszerek tervezésekor ezt a tényt fokozottan figyelembe kell venni, mivel alapvető minőségi jellemzőjük az ember-gép kapcsolat, az ún. „humán interfész” megfelelő kialakítása. A legfontosabb szempontok:

- Milyen beviteli módszer, beavatkozás a letermészetesebb a kezelőnek?
- Milyen megjelenítési, visszajelzési módszert a legcélszerűbb alkalmazni (vizuális, akusztikus, mechanikus jelzések)?

— A kezelő hogyan kap jelzést a rendszer hibás működéséről?

— Melyek a kezelő által elkövetett legvalószínűbb hibák?

— A kezelő beavatkozására adott rendszerválaszok egyértelműek?

— A kijelzési képet a kezelő könnyen olvashatja és érthető?

Ezek tisztázása után létrejön a tervezés alapjául szolgáló rendszerterv, amiből kiindulva elkezdhető a rendszer hardverjének a megtervezése.

Az áramköri tervezés

A megválaszolandó kérdések:

— Mennyi RAM/EPROM memória szükséges?

— Hány be/kimeneti áramkört kell használni?

— Milyen egyéb kiegészítő áramkörök szükségesek az adott feladat megoldásához?

A hardver elkészítése meglehetősen eszköz- és munkaiényes.

Lépései:

— a hardver megtervezése, a kapcsolási rajz elkészítése,

— a rajz alapján a nyomtatott áramköri lemez elkészítése,

— az alkatrészek beültetése a lemezre,

— a kész hardver bemérése, az egyes funkciók helyességének ellenőrzése.

A fejlesztési munkának ettől a részé-

től megkímél, azaz hogy sok olyan komplett, mikroprocesszoros rendszert tartalmazó bemért kártya kapható, ami önálló mikroszámítógépként használható. Ezek RAM és EPROM memóriát, valamint jó néhány be- és kimenetet tartalmaznak, ami szerint eligazodhatunk a megfelelő kiválasztásában. Valószínűleg a feladatunkhoz pontosan illő kártyát nem fogunk találni. A kiválasztáskor ezért kompromisszumokat kell kötni, és előfordulhat, hogy néhány áramkörü rész esetleg külön kell megépítenünk. Egy ilyen — talán legegyszerűbb — kártya leírása a magazin 1987/8. számában található: 8 be- és 8 kimenete van, maximum 64 kb-ot tartalmaz. Egy másik, univerzális használatú kártya tervét a sorozat egy későbbi részében közöljük.

A PROGRAM TERVEZÉSE

Ebben a lépésben a részletesen megfogalmazott feladatot programként realizáljuk. Bár a cikksorozat címe a hardverrel, az áramkörökkel kapcsol-



tos ismeretek bemutatására utal, most a teljesség kedvéért a programtervezésről, mint a rendszertervezés fontos lépéséről kissé részletesebben is szólnunk kell. A programtervezésben a következő módszereket használják: a folyamatábra-készítést, a moduláris programozást, a strukturált programozást és a felülről lefelé tervezést.

Az első kettő az assembler programozásnál használható jól, az utóbbiakat pedig magas szintű programozásokon (Pascal, C) alkalmazhatják.

Mielőtt ezeket a módszereket összefoglalnánk, soroljuk fel a programtervezésnél használt legfontosabb alapelveket:

— Célszerű kis lépésekben haladni és fejleszteni.

— A nagyobb részfeladatokat kis, egymástól logikailag elkülönülő modulokból kell felépíteni, mivel ezek önállóan tesztelhetők, és esetleges megváltoztatásuk nem igényli a teljes rendszer újratervelését.

— A feladatnak megfelelő programvezérlés lehetőleg egymás utáni, egyenként végrehajtható részekből álljon, és ne ide-oda ugrásokból, áttekinthetetlen programhurkokból, ciklusokból. Ez a hibakeresést is megkönnyíti.

— Célszerű minél több grafikus, vizuális leírást alkalmazni (a folyamatábra módszernek ez az előnye).

— A megfogalmazások, fogalmak egyszerűek és világosak legyenek.

— Olyan algoritmusokat kell felhasználni, amelyek ismertek és többszörösen kipróbáltak.

— A programtervezéskor figyelembe kell venni azt, hogy melyek azok a tényezők, paraméterek, amelyek megváltozhatnak.

— A kódolást csak a programtervezés teljes befejezése után szabad elkezdeni.

A folyamatábra módszer

Nagyon jól alkalmazható módszer. Legkiválóbb tulajdonsága az, hogy a programot grafikusán lehet megtervezni, és ezáltal jól láthatók az összefüggések és a programrészek közötti kapcsolatok.

Előnyei ezenkívül még, hogy — szabványos szimbólumokkal lehet dolgozni, így széles körben használható,

— a folyamatábrák megérthető programozási ismeretek nélkül is,

— a folyamatábrával mind a teljes program, mind az egyes részeit leírhatók.

Hátrányai:

— A folyamatábrát bonyolult meg-

erezni, megrajzolni, változtatni — újra kell rajzolni —, kivéve a legegyszerűbb eseteket.

— Nincs egyszerű módszer a folyamatára tesztelésére.

— Annak eldöntése, hogy részletes legyen-e a folyamatára, nem egyszerű. Ha túl tömör, nem adja vissza a program minden fontos részletét, ha pedig túl részletes, akkor nagyon nehéz a program áttekintése.

— A folyamatára csak a program szervezését mutatja, nem mutatja az adatszervezést és a be/kimeneti modulok felépítését.

— A folyamatábrában könnyű a nyilatkat ide-oda húzni, aminek az a következménye, hogy a kódoláskor sok hurkot, ide-oda ugrásokat kell megvalósítani.

Moduláris programozás

Az a módszert, amikor a teljes programot részekre — alprogramokra vagy más néven modulokra — osztjuk, moduláris programozásnak nevezzük. Ennek a módszernek az alapvető problémája az, hogy a tervező hogyan ossza fel a programot modulokra és miként egyesítse a modulokat komplett, működő programmá. E módszernek az előnyei:

— Az egyes modulokat könnyű megírni, tesztelni, a bennük lévő hibákat megkeresni.

— Egy modul valószínűleg sok helyen is felhasználható — más programokban is —, mert többnyire elég általános. Célzerű egy modulkönyvtárat létrehozni.

— Több programozó is könnyen együttműködhet a különböző modulok megírásánál.

— Egy-egy hiba általában egy modulhoz kapcsolódik csak, így könnyebb kijavítani.

Hátrányai:

— A modulok egymáshoz illesztése gondokkal járhat, különösen ha különböző programozók írták a modulokat.

— A modulok nagyon pontos és részletes dokumentációt igényelnek, mert a program más részeire is hatással vannak.

— A modulok tesztelése és a hibakeresés elég nehézkes lehet, mivel az egyes modulok olyan bemenő adatokat igényelnek, amelyeket más modulok állítanak elő. Ezért az ilyen modulok kipróbálásához külön olyan programrészeket kell írni, melyek az adott modulnak mintaadatokat generálnak. Ez járulékos — és valószínűleg nem szükséges — programozást igényel.

— A modulok gyakran több futás-

időt és memóriahelyet igényelnek, mivel hogy a modulokon belül gyakran ugyanazok a részfeladatok megismétlődnek.

Strukturált programozás

Ennél a módszernél — mint ahogy a neve is utal rá — a programokat a programozó alapstruktúrákból építi fel. Nevezünk egy utasítást vagy utasítássorozatot S-nek. A jelölés felhasználásával a következő alapstruktúrák adódnak:

1. Szekvenciális struktúra: S1, S2, S3 ... struktúrák egy programban sorban egymás után hajtódnak végre.

2. Feltételes struktúra: ha C igaz, akkor S1, különben S2. Itt C egy feltételt jelent.

3. Hurokstruktúra: amíg C igaz, csinálj S-t. Itt C egy feltételt jelent.

4. Indexstruktúra: I esetén S0, S1, ..., Sn, ahol I az index és 0, 1, ..., n értékű lehet. Ha I = 0, akkor S0, ha I = 1, akkor S1, ..., ha I = n, akkor Sn hajtódik végre.

Matematikailag bebizonyítható, hogy e négy struktúra felhasználásával tetszőleges program felépíthető. Ennek a módszernek az előnyei, hogy

— a műveletek sorrendjét könnyű követni, ezért könnyű a hibát megkeresni és tesztelni,

— az így felépített program már önmagát dokumentálja, tehát könnyen olvasható,

— ezt a módszert használva megnövekszik a programozói teljesítmény.

Hátrányai:

— Az ilyen programírást csak néhány magas szintű programozói nyelv (például PL/M, PL/Z, Pascal) támogatja utasításaival. Az assemblerben programozónak külön át kell alakítania egy ilyen módon megírt programot, de előnyösen használható dokumentációs célokra.

— A strukturált program általában hosszabb futásidőjű és több memóriát foglal el, mint a nem strukturált változat.

— A korlátozott számú struktúrával alkalmanként nagyon nehéz és bonyolult a feladatot megvalósítani. Az, hogy bármely program ezekkel a struktúrákkal megírható, igaz, de nem biztos, hogy a megírt program hatékony, kis tárgényű és gyors lesz!

— Az egymásba ágyazott „ha ... akkor ... különben” struktúrák esetleg nehezen olvashatók.

Végül is a strukturált programozást akkor ésszerű használni, ha programunk elég nagy — több mint 1000 uta-

sítás —, a futásidő és a memóriafelhasználás nem kritikus követelmény.

A memóriák olcsóbbá válása, valamint a nagyobb sebességű mikroprocesszorok megjelenése és elterjedése mind a strukturált programozás szélesebb körű felhasználásának irányába hat, mert jól karbantartható, technológiázható programozást tesz lehetővé.

Felülről lefelé programozás

Amikor a struktúrákból vagy modulokból a programot össze akarjuk állítani, felmerül az a kérdés, hogyan teszteljük az egyes részeket és miképpen, mennyire hatnak egymásra. A szokásos eljárás az, hogy az egyes részeket külön-külön, sokszor fáradtságos munkával teszteljük, majd a részek egymáshoz illesztése (az integrálás) a végére marad. Az a módszer, ami a tényleges programkörnyezetben lehetővé teszi a tesztelést és hibakeresést, a felülről lefelé programozás.

A módszer lényege, hogy először a teljes, átfogó programot írjuk meg, a benne szereplő egyes feladatokat, eljárásokat csak kijelöljük — egy nevet adunk nekik —, és csupán definiáljuk a név mögött álló feladatot. A következő lépésben ezeket a feladatokat — ha szükséges — ismét alapfeladatokra bontjuk, ismét definiáljuk azokat és így tovább. Mikor a leltábsában eljutottunk az elemi szinthez, onnan kezdjük a program megírását. Innen kezdődik a felfelé való programírás és a minden szinten elvégezhető tesztelés.

A módszer alkalmazásával olyan programunk lehet, melynek hatékonysága bizonytalan. Másrésztől hátránya az is, hogy a meglévő programokat, rutinokat nem biztos, hogy könnyen felhasználhatjuk, ugyanakkor a módszer nem eredményez általánosan felhasználható modulokat sem.

A fentiekben röviden összefoglaltuk az assembler programok fejlesztésénél felhasználható programtervezési módszereket. Nem adtunk egy kizárólagosan jó, „csak ezt használj” módszert, mert ha ilyen lenne, nem kellett volna a kérdéssel ennyit foglalkozni. Az egyes módszereket azért mutattuk be, hogy a feladat megoldójának ötleteket, módszereket adjunk, melyekből a feladat megoldása közben meríthet. Az igazi programtervezésnek világosnak, (mások által is!) érthetőnek, áttekinthetőnek kell lennie, amit csak sok és pontos munkával, gyakorlással érhetünk el.

BÖRZE.



COBRA
ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET
1097 Budapest, IX. Illovs ut 7. Telefon: 476-160/388

SZÁMLAKÉSZÍTÉSTŐL A KÖNYVELÉSIG

- COBRA—CONTO ügyviteli programrendszer moduljai
- Számlakészítő, Számlanyilvántartó, Bér- és jövedelemszámfejtő, Főkönyvi könyvelő, Anyagnyilvántartó, Általános könyvelési

PROGRAMOKI
Árak: 19 000—49 000 Ft-ig
KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

TUTTI

ELECTROCOOP
KISSZÖVETKEZET

- IBM PC kompatibilis gépek
- HARDVERTELEPÍTÉS
- SZERVIZ ÉS GARANCIA
- SZOFTVERES TÁMOGATÁS
- RÖVID HATÁRIDŐ

Cím: Bp., Üllői út 81. 1091
Tel.: 334-354



ENET

Lokális hálózat
IBM kompatibilis gépekre
Külön hardver nem szükséges
Ára gépként: 19 900 Ft + áfa
ECOSOFT KISSZÖVETKEZET
Tel.: 863-677

ADATÁTVITEL!



MIKROPO
BUDAPEST

Komplex számítástechnikai szolgáltatás!

Iroda:
VI., Nagymező u. 51.
Tel.: 325-768

ASY

Telefon:
415-166

Kereskedelmi és Szoftveriroda
1061 Bp., Liszt F. tér 10.
Telex: 22-4378

- ASY—16 SZUPERMIKRO számítógép (12 terminál, VME busz, UNIX)
- ASY—16/XT IBM kompatibilis mikroszámítógép
- ASY—16/AT IBM kompatibilis mikroszámítógép
- különleges igényeket kielégítő billentyűzetek
- szoftverfejlesztések



PERIFÉRIA

Elektronikai Fejlesztő és Szolgáltató KISSZÖVETKEZET
Bp. VII., Peterdy u. 30.
Telefon: 213-588

AJÁNLATA:

- P—XT: 140 E Ft-tól + áfa
 - P—AT: 200 E Ft-tól + áfa
- igény szerinti konfigurációk
- FX—1000 Printer: 75 E Ft + áfa
 - 40 MB-os WINCHESTER: 86 E Ft + áfa

procontrol



Kisszövetkezet
megnyitotta

COMPUTER SZAKÜZLETÉT

- hardver, szoftver
- elektronikai elemek
- integrált áramkörök
- számítógépek, perifériák
- blokkolóórák
- vonalkódeshozók, biztonsági rendszerek

SZEGED
Kazinczy u. 8. Tel.: 62/12-259
Bersenyi u. 2. Tx.: 82-726



BROTHER AX 15 típusú,
margarétafejes,
elektronikus írógép

Megvásárolható:
Budapest VI.,
Népköztársaság útja 2.
Tel.: 531-231



1146 Bp., AJTÓSI DÜRER
SOR 10.
Levélcím:
1393 Pf.: 319.
Telefon: 421-974
Telex: 22-6544

SZAKÜZLETEK:

1. Bp. VI., Szinyei Merse u. 1.
Tel.: 127-628. Tx.: 22-6684
2. 9022 Győr, Lukács S. u. 18.
Tel.: 96-14808. Tx.: 02-4679
3. 3100 Salgótarján, Ady Endre u. 1.
Tel.: 32-10971. Tx.: 22-9380

Sorozatunkban azokat az új hardver- és szoftvertermékeket ismertetjük, amelyek várhatóan általánosan elterjednek, és meghatározó szerepük lesz a fejlődés irányainak kialakításában.

Texas IC-k

A Texas cég az egyik legnagyobb — ha nem a legnagyobb — név a félvezetőgyártásban. Fejlesztései a tranzisztorgyártás megkezdésétől, az első IC-ken keresztül a mai grafikus szuperalkatrészekig, mindig a technológiai élvonalat képviselték. Ezenfelül a japánok előretöréséig a világ legnagyobb félvezetőgyártója volt, jelenleg is az első három között van.

A borító belső oldalán látható annak a TI 34010 grafikus processzornak a tömbvázlata, amellyel nagy felbontású, nagy sebességű, sok szint előállító kártyákat terveztek, amelyek ma már széles körben elterjedtek. Miből áll és milyen is ez a kártya?

Egy 32 bites CMOS mikroprocesszor van benne, grafikus rendszerekhez optimalizálva. Különlegessége — az alkalmazási célnak megfelelően tervezve — a bitenként címzhető, 1 Gbit címtérület, a képernyővezérlő. A cég most továbblépett. Kiegészítette a TMS340 családot. Előrelépett a kiegészítőkkel, mint például az 1. és 2. ábrán (hátsó borító) látható TMS34061 típusú videovezérlővel és a TMS34070 típusú színes palettával. Az előbbi bármilyen általános célú mikroprocesszorral együttműködhet, és kiküszöböli azt, hogy a szöveget és grafikát csak külön lehessen kezelni. Irányítani képes bármilyen monitort és 64 k/256 k video-, valamint ugyanakkora dinamikus tárat. A színes

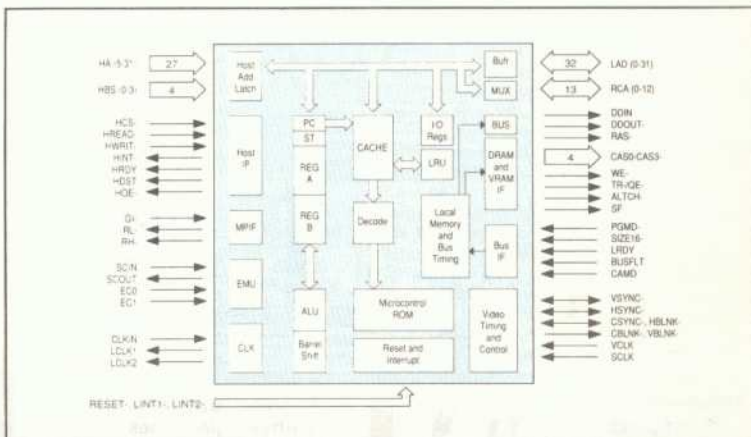
Merre tart a világ?

Alkatrészek, kártyák

paletta 4096 szín közül választhatja ki a képernyőre vitt 16-ot, és 1024 x 768 pont megcímzését teszi lehetővé.

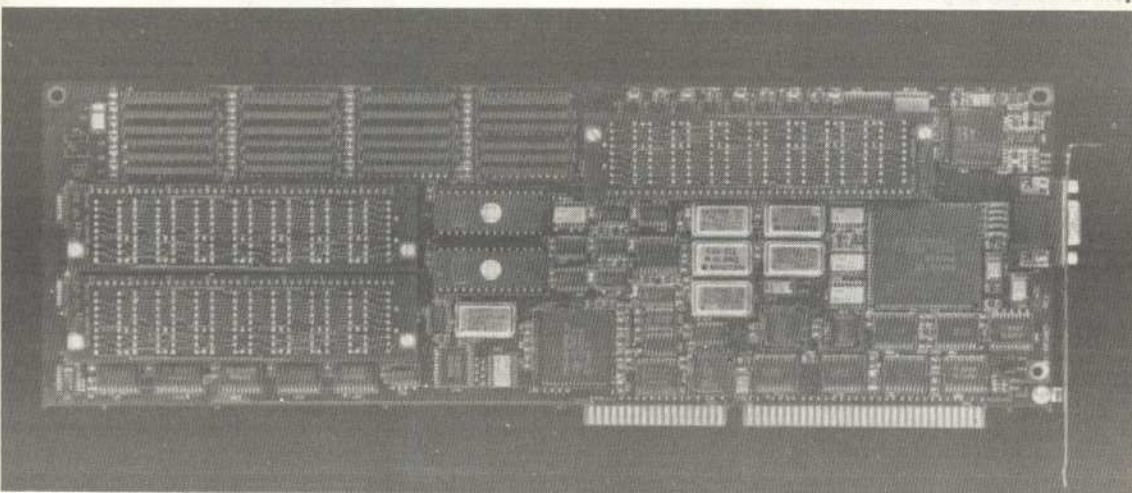
Ezenkívül tökéletesítették a grafikus processzorokat, aminek eredményeképpen egy új processzort hoztak létre, a TMS34020-as típust (3. ábra). Ez mintegy ötvönszer gyorsabb, mint az elődje, azzal felülről kompatibilis, multiprocesszoros üzemmódban dolgozhat más hasonló vagy egyéb processzorokkal. A

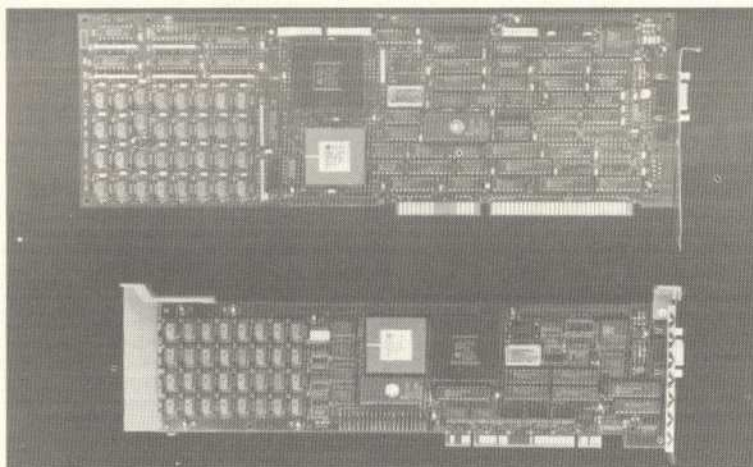
TMS34020-ast kiegészítették a TMS34082 típusú lebegőpontos processzorral (4. ábra), amivel 40 Mf lop érhető el (ez tízszerese a PC-koprocesszorokénak!). A TMS44C251 CMOS videotár (5. ábra) 1 Mbit kapacitású, 4 bites szervezésű. A három új alkatrészrel és néhány kiegészítővel készült 1024 x 1024, egyenként 8 bites színes képpontú rendszer blokkvázlatát, mint az egyik legfontosabb alkalmazási példáját mutatja a 6. ábra.



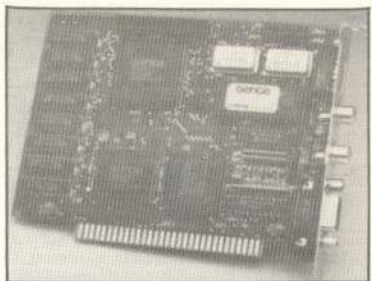
3. ábra

1. kép





2. kép

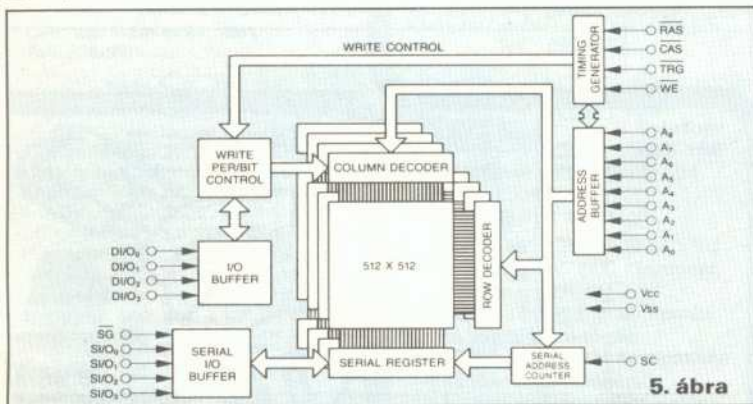


3. kép

dise — a Western Digital leányvállalata — PVGA1A IC-re egy Verticom MX elnevezésű sorozatot dolgozott ki (1. kép). Az AT kompatibilis kártyák 1024 × 768 darab 16 vagy 256 színű pontot kezelhetnek. A kártyán AutoCAD meghajtó is található a CAD-alkalmazások megkönnyítésére. A kártyákkal elsősorban a jelenleg mintegy százezer darabos CAD-piacot szeretnék meghódítani, de remélik, hogy a jövőre várható mintegy három és félszeres eladások is főleg az övék lesznek. (A piaci adatok a Dataquest nevű piackutató cégtől származnak.)

A még nagyobb igényű — és pénzü — vásárlóknak készült a cég Verticom HX sorozatának két új tagja (2. kép). A kártyák AT, illetve MCA kompatibilisek, maximálisan 2 Mbajt RAM tárat tartalmaznak, és hogy mit képesek nyújtani, az a hátsó borítón látható. A TwinFocus nevű eszközzel az AutoCAD-használók, ahogy ezt a kép is mutatja, egyidejűleg két interaktív, nagyítható ablakkal dolgozhatnak.

A cég Paradise leányvállalata a kispénzű felhasználók számára adott új megoldást a borító belső oldalán látható VGA Plus 16 kártyával. Ennek felbontása megegyezik az IBM VGA előírással (erről sorozatunk egy korábbi részében már volt szó). Különleges tulajdonsága, hogy megvizsgálja a számítógéprendszer egyéb részeinek konfigurációját, és amit talál, ahhoz alkalmazkodik.



5. ábra

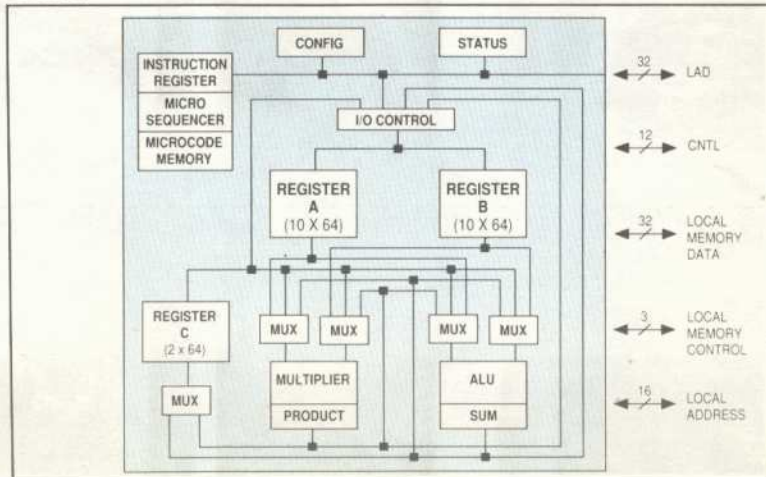
Motorola nyomtató-vezérlők

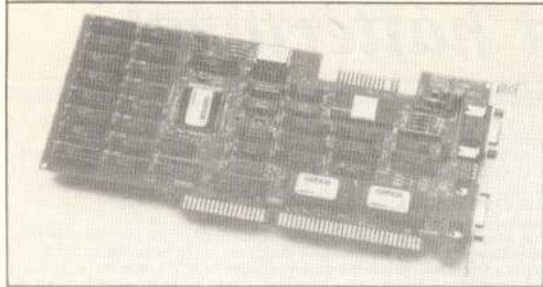
A cég kétmikronos, kétrétegű fém- és egymikronos háromrétegű CMOS technológiájának alkalmazásával két lézernyomtató-vezérlő IC-t hozott létre. Az LPC1 mikroprocesszor-illesztő, dinamikus RAM-kontrollert és DMA videoillesztőt, az ALPC1 ezenfelül a raszterműveletekhez kiegészítőt is tartalmaz (7. ábra). A mikroprocesszor a 680 × 0 sorozat lehet, a tár 8 Mbajtig 256 kbites vagy 1 Mbit-es tokokból állhat. A cég együttműködve a Personal Computer Products Inc. nevű vállalattal, komplett lézervezérlő kártyákat (8. ábra) is gyárt. Ezek képesek emulálni a HP LaserJet II-t, az Epson FX-80-at, a HPGL 7475A-t (plotter) és kompatibilisek a PostScript lapleíró nyelvvel.

Western Digital kártyák

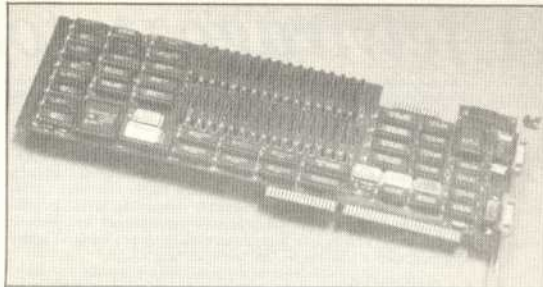
A cég a már említett T134010-re és a sorozat egy korábbi cikkében tárgyalt Para-

4. ábra

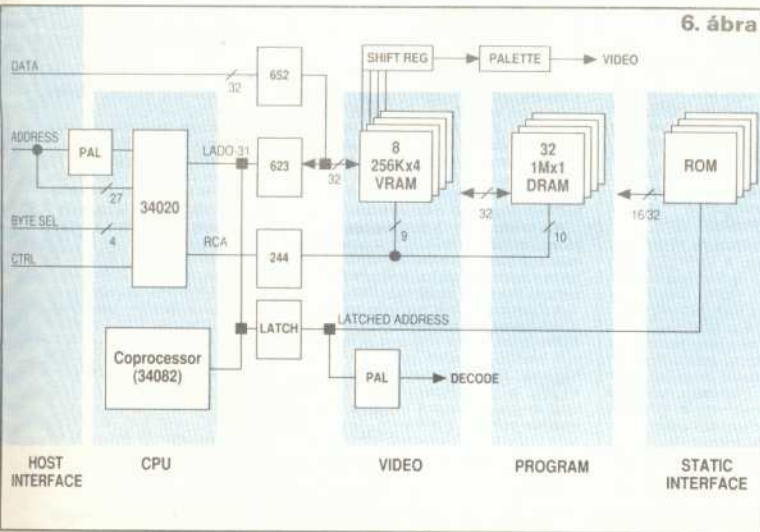




4. kép



5. kép



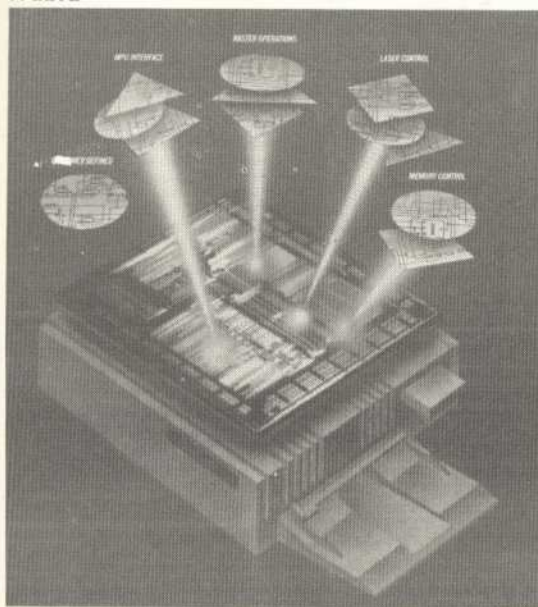
6. ábra

GENOA kártyák

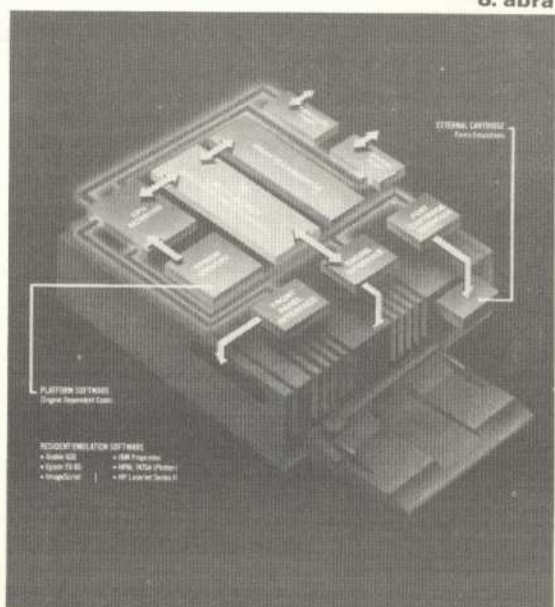
Az öt éves cég a PC-piacon az aránylag jó minőségű és olcsó grafikus kártyáiról ismert. Ő kezdte el például a 800x600 pont felbontású EGA kártyák gyártását. Ezt, az azóta is egyre szélesebb körben használt ASIC (Application-Specific Integrated Circuits – alkalmazásspecifikus IC) technológiának köszönhetők. Új termékek a SuperEGA HiRes+ (3. kép) lehetővé teszi, hogy monokrom monitorral EGA és EGA monitorral Hercules kompatibilis szoftver dolgozhasson együtt. A SUPERVGA 16-Bit Adapter (4. kép) 16 bites busz és saját RAMBIOS használatával megkétszerezi a VGA-termékek működési sebességét. Végül a SuperHGA (5. kép) kártyán levő RISC processzorral (erről a típusról korábban már írtam) mint egy százszor gyorsabb tehetők a VGA funkciók, és egyúttal sokkal egyszerűbbé teszik a működtető szoftvert.

Simonyi Endre

7. ábra



8. ábra



A felvirágoztatott háttértároló

A leggyakrabban használt, például szövegszerkesztő, assembler, monitor programoknál előnyös, ha ezeket gyorsan és egyszerűen tudjuk a számítógép memóriájába tölteni és futtatni. Azoknak a számítógépeknek, melyeknek lemezmeghajtójuk van, nem gond, de nagyon sok személyi számítógép tulajdonosának még nincs ilyen meghajtója, hanem háttértárolónak kezeltés magnetofont használ. Ez az eszköz azonban viszonylag lassú, és sokszor jogos túrelmentlenül várjuk programunk betöltésének befejezését. Hibás betöltéskor pedig kezdhetjük az egészet előlről. Jogos tehát az igény egy olyan eszközre, ami a következő feltételeknek tesz eleget:

- tárolja a programokat,
- egyszerű vele a betöltés,
- minél kevesebb vezetékekkel kapcsolódik a számítógéphez,
- a legtöbb személyi számítógéphez kapcsolható, azaz típusfüggetlen,
- az ára nem magas.

Az ezeket a feltételeket kielégítő eszközt Rögzített Tartalmú Soros Tárolónak, a továbbiakban RTST-nek nevezzük. Tárolóeszközként nagy kapacitású EPROM memóriát alkalmazunk — például 16 kbájtos 27 I28-as, 32 kbájtos 27 256-os vagy 64 kbájtos 27 512-es típust —, ezek ára folyamatosan csökken. Az egyes személyi számítógépekhez kapható bedugható („plug-in”) modulok is ilyen memóriát tartalmaznak. Az RTST-ben új a tartalom kiolvasásának a módja: a memóriából való kiolvasást nem párhuzamosan, hanem sorosan, bitenként valósíthatjuk meg. A sorosan érkező bitekből a bájtokat a számítógépben lévő program rakja össze. A következőkben leírt megoldás már nem általános, hanem a Primo számítógépre készült, de a leírás alapján más számítógépekre is adaptálható. A Primóhoz készült RTST kapcsolási rajza (0,5 Mbájtt) az ábrán követhető végig.

A kapcsolási rajzon szereplő ötpólusú csatlakozó bekötése a Primo botorkmány (B) csatlakozójának bekötésével egyezik meg. A törölő (RSTBE) bemenetre adott jellel alaphelyzetbe hozzuk a V10, V11 és V14 jelű számlálókat. Ezután kiolvaszuk a (DKI) kimeneten keresztül a V15-ös multiplexer Y adatkimenetét, ami az EPROM nulladik címén lévő bájt nulladik bitje. A léptető bemenetre (CLBE) adott jellel léptetjük a V14-es felső 8-as számlálóját, és ismét kiolvaszuk a multiplexer kimenetét, ami a megcímzett bájt első bitje. A nyolcadik léptetőskor a multiplexert címző számláló ismét alaphelyzetbe kerül, azaz ismét a megcímzett bájt nulladik bitje lesz a kimeneten. Azonban az átvitelen megjelenő jel a V11, V10-es címszámláló láncot is lépteti, és a megcímzett I. címen lévő bájt jelenik meg az EPROM adatkimenetén. A kiolvasást ekként ismételve, sorban egymás után, bitenként kiolvasódik az EPROM tartalma. Az első EPROM-tokban lévő utolsó bájt kiolvasása után a V14-es IC also számlálóját is egyet lép. A számláló kimenete a V13-as dekódolóra kapcsolódik, melynek kimenetei sorban egymás után kiválasztják az EPROM-tokokat, ebben az esetben a második

kat. A kiolvasás tehát a következő tokból folytatódik. A V13-as 12. lábára adott H szintű jellel az egész kártya működését tilthatjuk.

Az RTST a számítógéptől két kimeneti és egy bemeneti bitet igényel. A kapcsolás maximum nyolc darab 64 kbájtos tokkal használható, azaz a tárolási kapacitás maximum 512 kbájtt lehet! Természetesen kisebb kapacitású EPROM-ok is használhatók, a JI jelű átkötésmező megfelelő összekötéseivel.

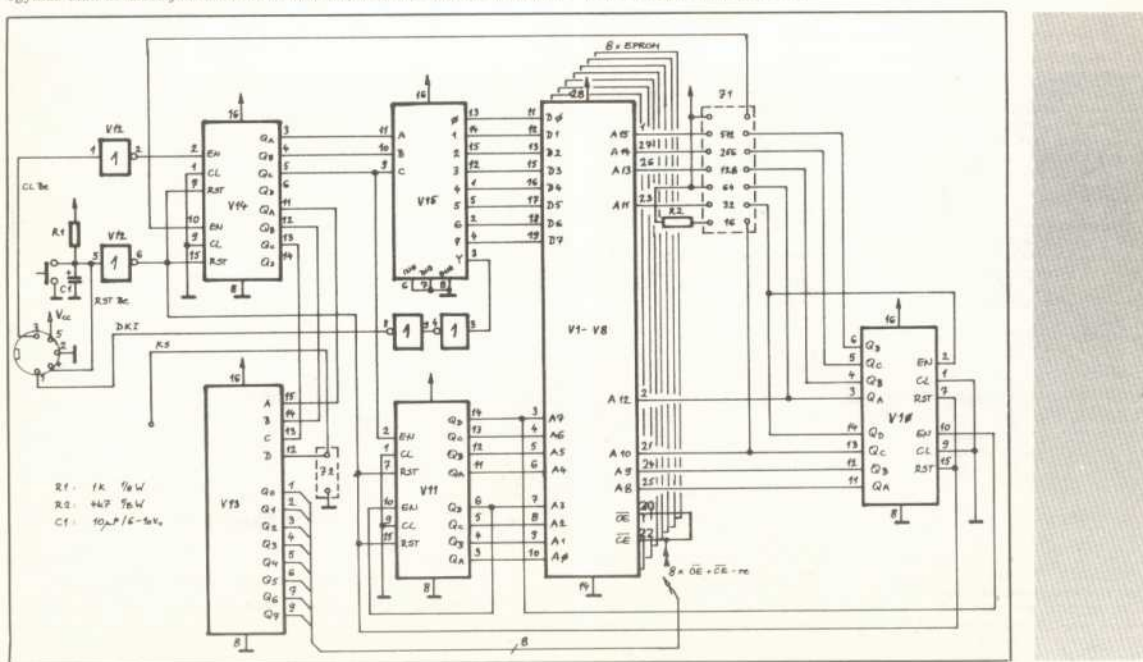
A következő lépés az RTST-t működtető szoftver megtervezése és megvalósítása. A szoftver legfontosabb eleme egy olyan betöltőprogram, amely képes a fenti hardver működtetésére, azaz az EPROM tartalmát a memóriába adott kezdőcímetől képes betölteni, majd a betöltött programot elindítani. Az EPROM tartalmát célszerű két részre bontani: az első rész tartalmazza az EPROM második részében található programokra vonatkozó információkat, azt hogy:

- az EPROM melyik címén kezdődik a program (EKCIM),
 - a memória melyik címétől töltődik a program (MKCIM),
 - melyik memóriacímig tart a betöltés (MVCIM),
 - a memória melyik címéről indul a program futása (ICIM).
- Ez a 4 x 2 bájt egyértelműen meghatározza a programot, és ezek alapján a program betölthető és futtatása elindítható.

Az első EPROM elején, a programokra vonatkozó paramétertáblázat előtt helyezkedik el az a program, amelynek feladata, hogy a felhasználó egy menüt fektalvá, kiválaszthassa a futtatni kívánt programot. Az így felépített rendszerrel elvileg tetszőleges program futtatható, mivel csupán az EPROM első részében lévő menü szövegét és a program paramétereit kell megfelelő módon beírni.

A következő — a fentiek alapján működő — program Primo számítógépre készült. Mivel a Primo botorkmány-csatlakozóját használtuk fel, ami csak egy kimeneti vonalat tartalmaz, ezért az áramköröket törölő RESET jelet egy nyomógombbal kell generálni. Ez azt jelenti, hogy bekapcsoláskor vagy új program betöltése előtt a gombot meg kell nyomni.

A Primóra készült betöltőprogram két, fizikailag is különálló részből áll. Az első program a tulajdonképpeni fizikai betöltő, amely az RTST első EPROM-jának elején lévő menüt megjeleníti, valamint a kiválasztott programot betöltő és indító programot tölti be a képernyő-memória utolsó 256 bájtjába. Ez a tartomány 64-es gépeknél az FF00H, a 48-as gépeknél a BF00H, a 32-es gépeknél a 7F00H memória kezdőcímen található. Azért esett a választás erre a területre, mert egyszerűen a futtatandó program kiválasztása után ez a program már „megsemmisülhet”, másrészt így módon nem foglalkalunk el a memóriából olyan részt, ami valamelyik program felhasznál. A fizikai betöltőnek a számítógép memóriájában kell lennie azért, hogy működtesse az RTST kártya áramkörét. Primónál a fizikai betöltő kétféleképpen helyezhető el:



Programnév: RTSTOL Programhossz: 38 bajt

A program feladata: RTST programok betöltése

Felhasználó regiszterek: A, BC, DE, HL, IX

```

1 :ROMBAH LEVS BETOLTO
2 007B      ORG 0710H
3 007E      D1 00 FF   RTSTOL: LD HL,0FF00H :MKCIM
4 007E      11 FF FF   LD DE,0FFFFH :MVVICM
5 0081      FD 21 00 FF  LD IX,0FF00H :ICIM
6 0083      06 00      TOLT: LD B,B :EDY HAJT BEOLV.
7 0087      0B 40      IDE: LD A,(40H) :OLVAJAS
8 0089      1F      RRA
9 008A      CB 19      RR C
10 008E      3E 0D      LD A,9DH :SZAMLAGO LEPTEKES
11 0090      07 00 00  OUT (40H),A : 0 SZINT KI
12 0090      3E ED      LD A,0EDH
13 0092      D3 00      OUT (08H),A : 1 SZINT KI
14 0094      10 F1      DJNZ IDE : HA VAN MEG A BAJTOL
15 0096      71      LD (HL),C
16 0097      23      INC HL : KOVETKEZO BAJT
17 0098      E5      PUSH HL
18 0099      AF      XOR A
19 009A      ED 52      SBC HL,DE :UTOLSS BAJT VOLT?
20 009C      E1 08      POP HL
21 009D      30 E0      JR C,TOLT :HA NEM, VISSZA
22 009F      FD E0      JP (IX) :OLRAS ICIM-RE
23 00A1      END
    
```

Programnév: RTSTPR Programhossz: MAX.256 bajt

A program feladata: Az RTST directory betöltése és a kiválasztott program indítása

Felhasználó regiszterek: A, BC, DE, HL, IX

```

1 :RTST PROGRAMOK BETOLTOJE
2 0000      ORG 0H
3
4 FF60      MENU: EQU 0FF60H :SZOVEG KEZDOTE
5 FF60      TABL: EQU 0FF60H :TAAH, TAAJAZAT KEZDOTE
6 0015      DSFL: EQU 15H :PRIMO KEPEKYOKEZELO
7 0025      KLAV: EQU 25H :PRIMO BILLENYOKEZELO
8 0085      STOLT: EQU 0085H :FIZIKAI BETOLTO A ROMBAH
9
10 0000      21 60 FF   KIIR: LD HL,MENU
11 0003      7E      CLR B
12 0004      FE FF   CP 0FFH :SZOVEG VERGE?
13 0006      20 00   JR C,0015 :IGEM
14 0008      CD 15 00  CALL DSFL :KIIRAS A KEPEKYOKE
15 000D      23      INC HL :UJ KAKARTEK
16 000E      10 F5   JR ISMET
17 000E      CD 25 00  (UJBI) CALL KLAV :VARAS BILLENYOKE
18 0011      20 F5   JR C,0011 :HA NEM NYOMTA
19 0013      D8 30   SUB 30H
20 0015      FE 00   CP 0H : SZAMBILLENYTY ?
21 0017      38 F5   JR C,UJBI : 0 ?
22 0019      FE 00   CP 00H :
23 001B      70 F1   JR C,UJBI : 0 ?
24 001D      07      TOLT: RLCA : * 2
25 001E      07      RLCA : * 4
26 001F      07      RLCA : * 8
27 0021      4F      LD A,A
28 0021      06 00   LD B,B :BETÖLTETÁJ TÁBLAJAZ
29 0023      21 00 FF  LD HL,TABL :HL:TÁBLAJAZAT KEZDOICIM
30 0026      09      ADD HL,BC :TÖRVÉNYES CIM
31 0027      4E      LD C,(HL)
32 0028      29      INC HL
33 0029      40      LD B,(HL) :BC=EKCIM
34 002A      29      INC HL
35 002B      78      LD A,B
36 002C      FE FF   CP 0FFH
37 002E      07 00 00  JR C,KIIR :ELHESZ HIRUJ PROGRAM
38 0030      E5      PUSH BC
39 0031      05      DEC B : MERT 256-OT NEM LEPTOKE
40 0032      C5      CPLEP: PUSH BC :FELLEPTETKES AZ EPROMBAH
41 0033      06 00   LD B,B : EDY BAJT
42 0035      3D 3D   CLEP1: LD A,9DH :SZAMLAGO LEPTEKES
43 0037      D3 00   OUT (0),A : 0 KIVITEL
44 0039      3E ED      LD A,0EDH
45 003B      D3 00   OUT (0),A : 1 KIVITEL
46 003D      10 F6   DJNZ CLEP1
47 003F      C1      POP BC :BC=EKCIM
48 0040      78      LD A,B
49 0041      B1      OR C :BC=0 ?
50 0042      0B      DEC B
51 0043      30 ED   JR NZ,CLEP1 :HA BC NEM 0
52 0045      7E      LD B,(HL)
53 0046      4E      LD C,(HL)
54 0047      23      INC HL
55 0048      46      LD B,(HL) :BC=MKCIM
56 0049      7E      INC HL
57 004A      46      LD C,(HL)
58 004B      23      INC HL
59 004C      56      LD D,(HL) :DE=VICM
60 004D      23      INC HL
61 004E      C5      PUSH 2PC
62 004F      C1      LD C,(HL)
63 0050      23      INC HL
64 0051      46      LD B,(HL) :BC=ICIM
65 0052      C5      PUSH BC
66 0053      FD E1   POP IX :IX=ICIM
67 0055      E1      POP HL :HL=MKCIM
68 0056      CD 85 80  JP STOLT :FIZIKAI BETOLTO ROMBAH
69
70 0000      ORG 60H : MENU SZOVEGVE
71 0000      DE 20H :RTST PROGRAMOK...:1,10H
72
73 0070      0D 00 3D 3D 3E  DB 00H,00H,0=EDI
74 0071      45 44 49      DB 00H,1=1F,00H
75 0072      0D 31 3D 3E 50  DB 00H,2=2EP,00H
76 0073      2D 42 55 47      DB 00H,3=3E,MAJ
77 0074      0D 32 3D 3E 45  DB 00H,4=4EP,00H
78 0075      50 28 50 52 47  DB 00H,5=5E,MAJ
79 0076      0D 33 3D 3E 53  DB 00H,6=6E,MAJ
80 0077      50 28 4D 41 53  DB 00H,7=7E,MAJ
81 0078      0D
82 0079      0D
83 0080      0D 01E0 4340 55  DB 0100H,4300H,5540H,5570H :EDI
84 0081      3055
85 0082      0D 18C0 430F 53  DB 1800H,4300H,530FH,4300H :CP,BUG
86 0083      C743
87 0084      0D 2400 846F E7  DB 2400H,8400H,875FH,8400H :EP,10G
88 0085      0084
89 0086      0D 2080 5770 58  DB 2080H,5700H,570FH,5700H :S2,MAJ
90 0087      0057
91 0088      END
    
```

1. Fizikai betöltő a BASIC ROM-ban

A Primo BASIC-et tartalmazó EPROM-tokokban van olyan „üres” hely, amit a BASIC nem használ, ide a fizikai betöltőprogram beírható. Ez azonban azzal a kényelmentességgel jár, hogy a Primóba beforrasztott EPROM-tokok közül az üres helyet tartalmazó tokot ki kell emelni, majd ismét vissza kell helyezni (jobbát a helyére foglalatot forrasztani). Egészen pontosan: az első 4 k memóriatarományt tartalmazó tok — a panel szelén lévő — 67BH-6A0H címére kell beírni az RTSTOL nevű betöltőprogramot.

2. Fizikai betöltő szalagról

Semmiféle fizikai beavatkozást nem igényel, ha a fizikai betöltőprogram szalagról betölthető öninduló assembler programmá alakítjuk. Ekkor a program a 4127H-414FH RAM-területre töltődik. A fizikai betöltő indítási címe 4134H. Ha ez a betöltőt program nem sérül meg, újra elindítható a call(16679) utasítással.

A fizikai betöltőprogram olyan a kialakítása, hogy mind a menükiró és -kiválasztó programot, mind a segítségéző kiválasztott felhasználói programot képes betölteni és elindítani. Ezt úgy értük el, hogy a fizikai betöltőt két részre tagoltuk: az első rész állítja be a betöltési és indítási paramétereket, a második rész pedig a betöltést és az indítást végzi. Először a fizikai betöltőt az RTSTOL címkén keresztül aktiváljuk, amivel a menü-programot betöltjük be és indítjuk el. A menüből kiválasztott felhasználói program betöltési paramétereit a regiszterekbe töltve, a vezérlés az STOLT címen keresztül ismét a fizikai betöltőnek adható át, ami betölti és elindítja a programot. A menüprogram három részre osztható. Az első rész az RTSTPR nevű program.

A program egyrészt a képernyőre írja a menüt, majd a felhasználói programot kiválasztó billentyű megnyomására (0-9 számjékok) várakozik. Ertekezhető kódd megadása után betölti az EKCIM paramétert. Mivel a számláló 0-ról indulnak (RESET után), ezért ha nem az első programot választottuk, akkor tovább kell a betöltenő program eljégre lépni az EPROM-okban (C5LEP címké). Ezután történik a többi paraméter betöltése, majd a vezérlés átadása a fizikai betöltőre.

A második rész tartalmazza a képernyőre kiírandó menüsöveget (TEXT címke). A szöveg végén az első FFH értékkel jelölti ki. A harmadik rész egy táblázat (TKCIM címke), ami a felhasználói programok paramétereit tartalmazza. A példaként négy programot tartalmazó táblázat az alábbi.

Programnév	EKCIM	MK CIM	MVCIM	ICIM
EDI	100	42E8	5548	5538
PBUG	1360	43C0	53BF	43C0
EPR.PRG	2400	E400	E75F	E400
SZMAS	2800	5700	5870	5700

(EDI = Editor-assembler, PBUG = Monitor-disassembler, EPR.PRG = 2716/2732 EPROM programozó, SZMAS = szalagmásoló program)

A szöveg és a táblázat még nem használt bajtjainak értéke FFH. Ezért újabb programokat úgy illesztünk be a rendszerbe, hogy azokat egyszerűen hozzáírjuk a menüsöveghez, és a program paramétereit beírjuk a táblázatba. A meglévő EPROM-tartalmak nem kell módosítani. Nagyobb kapacitású RTST alkalmazások kicsinek bizonyulhatnak a menüsöveg mezője, és a maximum tíz felhasználói program paramétereit befogadó táblázat is. Aggodalomra azonban nincs ok, mert mindkét korlát bővíthető úgy, hogy a menüprogram méretét 256 bajt hosszról nagyobbra — 512 vagy 1024 bajtra — növeljük.

Konverziós program

IBM PC-re

Az IBM PC gépeken máig sincs egységes, teljes magyar ékezetes karakterkészlet. Az eredeti IBM karakterkészlet tartalmaz ékezetes betűket, de néhány — például a nagy hosszú ó stb. — hiányzik.

Ezen csak a karaktergenerátor EPROM cseréjével lehet segíteni. A Videotvont VT110-esnél ezt meg is tették, amiólt a gép elvészttette az eredetiekkel való teljes kompatibilitását. Az is elképzelhető, hogy a hiányzó karaktereket más — megegyezőszerinti — karakterekkel helyettesítsük, például hosszú ó = Ö.

Sajnos a VT110 módosított karaktergenerátorában a hiányzó karaktereket olyan kódokra helyezték, amelyekből néhány még véletlenül sem hasonlít az eredeti IBM megfelelőjére, például a hosszú l = Ç [128]. Ezért ha VT110-en készült szövegfájl akarunk IBM gépen kinyomtatni, a fájlt konvertálni kell. Bemutatok a VT110 — IBM konverzióra egy megoldást.

A program a DOS (V2.0 vagy későbbi) Standard perifériakezelési lehetőségét használja ki. A DOS öt Standard egységet ismer:

Név	Rövidítve	Fájlszám	Alapértelmezés
Standard Input	STDIN	0	CON
Standard Output	STDOUT	1	CON
Standard Error	STDERR	2	CON
Standard Auxiliary	STDAUX	3	AUX
Standard List	STDLST	4	PRN

A DOS az STDIN-t, az STDOUT-ot és az STDERR-t használja a felhasználóval való kommunikációra. Mindegyik a konzolra vonatkozik. A CON bemenetként a billentyűzet, kimenetként a képernyőt jelenti. A másik kettő a printer (PRN) és az első soros vonal (AUX) kezelésére szolgál.

Ezek az eszközök mindig megtalálhatók a megadott fájlzámokon (0—4). Ugyanúgy kezelhetők, mintha fájlok lennének, írni-olvasni lehet őket.

Ezenkívül a felhasználónak lehetősége van arra is, hogy az STDIN és STDOUT más egységre vagy fájlra irányításával elérhesse például azt, hogy egy program kimenetét ne a képernyő-

VT110 — IBM karakterkonvertáló program

```

[vt.asm
[
[ VT110 — IBM karakterkonvertáló program
[
[ (c) 1988 Szinyei Gerzson KVM-EABT
[
DOS equ 23h ; DOS Funkciók interruptja
STDIN equ 0 ; Standard Input fájlneve
STDOUT equ 1 ; Standard Output fájlneve
READ equ 3fh ; Fn 3f-olvasás file-ból vagy egységrel
WRITE equ 40h ; Fn 40-írás file-ba vagy egységre
PRINT_STRING equ 09h ; Fn 09-karakterlánc kiírása
PR equ 02h ; Fn 02-karakterlánc kiírása
CR equ 13 ; Kocsi vissza
LF equ 10 ; Sorozatlás
EOF equ "ß" ; Szöveg vége a DOS PRINT_STRING-nél

Vtsegs segment "code"
assume C8:Vtsegs,D8:Vtsegs
org 100h

parmas: db 0 ;Paraméter sor hossza (PSF:00h)
org 100h ;.COM program miatt
pr: far
ebp byte ptr parmas,0 ;Van paraméter?

jne loop ;igen-program futás

mov di,offset text ;főm-bevezető szöveg kiírása
mov ah,PRINT_STRING
int DOS
jmp end_process ;Vissza a DOS-hoz

text: db "VT110 ==> IBM Karakterkonvertáló program",CR,LF
db CR,LF
db "Hívás: VT (bemeneti fájl)név kimeneti fájl)név",CR,LF
db EOF

loop: mov ah,READ ;Egyetlen karakter
mov ah,STDIN ;olvasása a Standard
mov dx,offset char ;Input-ról
mov cx,PR ;
int DOS ;
jc error ;OK?

inc eax
jne end_process ;LUF?

mov al,byte ptr char ;A beolvasott karak-
mov dx,offset Newtable ;ter konverziója a
int DOS ;Newtable" alapján
mov byte ptr char,al ;

mov ah,WRITE ;A konvertált karakter
mov ah,STDOUT ;írása a
mov dx,offset char ;Output-ra
mov cx,PR ;
int DOS ;
jc error ;OK?

cmp ax,1 ;LÉVEZ TELE?
jne error

```

```

jmp short loop

end_process: mov ah,C800h ;Visszatérés a DOS-hoz 0-Ah
int DOS ;ERRORLEVEL-lel (ok)

error: mov ah,C003h ;Visszatérés a DOS-hoz 1-eAh
int DOS ;ERRORLEVEL-lel (hiba)

char: db 0 ;karakter tároló
org 200h ;A Debug-gal, vagy
;PCTool-szal való könnyebb
;szerkesztés miatt

Newtable: ;A módosításokat tartalmazó
; tábla

db 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
db 16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
db " " " " " " " " " " " " " " " " " " " " " "
db "0123456789ABCDEF"
db "FGHIJKLMNOPQRSTUVWXYZ"
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "
db 0ch,"0000000",0e7h,93h,95h,98h,"AA" ;ezekben a sorokban
;van változás az er-
;edetivel képest

db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "

VT endp
Vtsegs ends
end VT

; Az eredeti karaktertábla

db 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
db 16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
db " " " " " " " " " " " " " " " " " " " " " "
db "0123456789ABCDEF"
db "FGHIJKLMNOPQRSTUVWXYZ"
db " " " " " " " " " " " " " " " " " " " " " "
db "00000000",0e7h,93h,95h,98h,"AA"
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "
db " " " " " " " " " " " " " " " " " " " " " "

```

re, hanem egy fájlban küldje és később szerkessze vagy nyomtassa.

Természetesen ez csak akkor működik, ha az adott program az STDOUT-ot használja kiíratásra, nem pedig ezt megkerülve, mondjuk közvetlenül ír a videomemóriába.

A Standard IO a DOS kijelentkező promptjánál irányítható át, a végrehajtandó program neve után beirt speciális karakterekkel. Ezek a karakterek: <, >, ,;.

Jelentésük:
< fájl vagy egységnevé. Az STDIN a billentyűzet helyett a megadott fájlból vagy egységből olvas.

> fájl vagy egységnevé. Az STDOUT a képernyő helyett a megadott fájlba vagy egységre ír. A fájl régi tartalma elvész.

»fájlnevé. Ugyanaz, mint > fájlnevé, de a fájlba hozzáfűzéssel (APPEND) ír.

prog1; prog2. Az első program STDOUT-ját egy ideiglenes fájlba írja az aktuális alkönyvtárban, azután innen veszi a második program STDIN-jét. A második program vége után letörli az ideiglenes fájl. Ezekről az átirányításokról a parancsfeldolgozó (COMMAND.COM) gondoskodik, a hívott program számára láthatatlanul. Például:

```
DIR »CATALOG          a directoryt hozzáfűzi a
                      CATALOG nevű fájlhoz.
```

A VT.COM készítése: az assembly listát írjuk be egy .ASM kiterjesztésű fájlba (VT.ASM). Ezután fordítsuk le az alábbi módon:

```
MASM VT;
LINK VT;
EXE2BIN VT.EXE VT.COM
```

Ezután már megvan a VT.COM fájl, amit az alábbi módokon használhatunk:

```
TYPE UZENET !VT > PRN
Az UZENET fájl tartalmát konvertálva kinyomtatja.
```

```
VT <forrás fájl neve > cél fájl neve
Ekkor a VT a forrásfájlt konvertálja és tárolja a célfájlban.
```

Ha a programot paraméterek nélkül futtatjuk, kiírja a helyes szintaxisát. Természetesen fájlnevek helyett adhatunk meg egységeket is, a kimeneti fájlnevé elhagyása esetén pedig automatikusan a képernyőre ír.

A bejelentkező szöveget mindenki tetszés szerint módosíthatja a forráslistában vagy a kész programban, ekkor azonban a szöveg hossza nem változhat.

Eredeti (VT110)		Konvertált (IBM)	
Kód	Karakter	Kód	Karakter
A0H	á	A0H	á
82H	é	82H	é
A1H	i	A1H	i
98H	ó	A2H	ó
94H	ö	94H	ö
8BH	ő	93H	ő
A3H	ü	A3H	ü
81H	ű	81H	ű
96H	Ä	96H	ü
8FH	Å	8FH	Ä
90H	É	90H	É
80H	Ì	8CH	ì
8CH	Ó	95H	õ
99H	Ö	99H	Ö
8AH	Θ	E9H	θ
97H	ù	97H	ù
9AH	Û	9AH	Û
8DH	Ÿ	98H	ÿ

Az ékezetes karakterek konverziós táblája

A VT program természetesen megfelelően módosítva más feladatok ellátására is képes (például kisbetű-nagybetű konverzióra). Ehhez a konverziós táblát a Debug rendszerprogrammal vagy a népszerű PCTools programmal módosíthatjuk.

A könnyebb módosítás kedvéért került a karaktertábla kerek (200H) címre a fájlban.

A Debuggal készült hexa dumpon láthatjuk a jelenlegi konverziós táblát. Az átirandó karakter helyére (200H + a karakter kódja) beírjuk a helyette látni kívánt karakter kódját.

Szinyei Gerzson

A) debug VT.COM		Hexa dump a karaktertábláról	
-d 200 2ff			
0FE0:0200	00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F	
0FE0:0210	10 11 12 13 14 15 16 17-18 19 1A 1B 1C 1D 1E 1F	
0FE0:0220	20 21 22 23 24 25 26 27-28 29 2A 2B 2C 2D 2E 2F	! " # \$ % & ' () * + , - /	
0FE0:0230	30 31 32 33 34 35 36 37-38 39 3A 3B 3C 3D 3E 3F	0123456789:;<=>?	
0FE0:0240	40 41 42 43 44 45 46 47-48 49 4A 4B 4C 4D 4E 4F	@ABCDEFGHIJKLMNO	
0FE0:0250	50 51 52 53 54 55 56 57-58 59 5A 5B 5C 5D 5E 5F	PQRSTUVWXYZ[\]^_	
0FE0:0260	60 61 62 63 64 65 66 67-68 69 6A 6B 6C 6D 6E 6F	`abcdefg hijklmno	
0FE0:0270	70 71 72 73 74 75 76 77-78 79 7A 7B 7C 7D 7E 7F	pqrstuvwxyz{ }~.	
0FE0:0280	80 81 82 83 84 85 86 87-88 89 E9 93 95 98 8E 8F	
0FE0:0290	90 91 92 93 94 95 96 97-A2 99 9A 9B 9C 9D 9E 9F	
0FE0:02A0	A0 A1 A2 A3 A4 A5 A6 A7-A8 A9 AA AB AC AD AE AF	
0FE0:02B0	B0 B1 B2 B3 B4 B5 B6 B7-B8 B9 BA BB BC BD BE BF	
0FE0:02C0	C0 C1 C2 C3 C4 C5 C6 C7-C8 C9 CA CB CC CD CE CF	
0FE0:02D0	D0 D1 D2 D3 D4 D5 D6 D7-D8 D9 DA DB DC DD DE DF	
0FE0:02E0	E0 E1 E2 E3 E4 E5 E6 E7-E8 E9 EA EB EC ED EE EF	
0FE0:02F0	F0 F1 F2 F3 F4 F5 F6 F7-F8 F9 FA FB FC FD FE FF	

Elektronok — elektronika

Az elektronika szóban egy elemi részecske neve húzódik meg: az *elektroné*. Miért játszik pont ez az elemi részecske olyan meghatározó szerepet mai számítógépeinkben, rádiókészülékeinkben, kvarcóránkban, telefonközpontjainkban — felsorolni is lehetetlen, még hol mindenhol? Például miért nem „*protonikus*” vagy „*neutronikus*” szerkezetek terjedtek el világunkban? A válasz egyszerű. Ennek az elemi részecskének számunkra kitüntetett *jel tulajdonságai* vannak:

— *tömege* igen kicsi ($\sim 9,1 \times 10^{-31}$ kg). Ez jó, mert felgyorsításához — mozgatásához — nagyon kevés energia szükséges;

— *kis tömege ellenére* igen „erős” — elektromos-mágneses — *kölcsönhatásban* van környezetével (töltése $\sim 1,6 \times 10^{-19}$ C). Emiatt jól érzékelhető, jól „terelhető”, energiátárolásra alkalmas;

— *sok olyan*, a nekünk fontos hőmérséklet-tartományban, ahol gépeink működnek, szilárd halmazállapotú anyag van — ezeket nevezzük *elektromos vezető anyagoknak* —, melyben az elektronok nagy tömegben, „szabaddan” (például a kristályszerkezetben nem kötötten) vannak jelen. És sok olyan anyag is — ezek szigetelők —, melyben ilyen „szabad” elektronok gyakorlatilag nincsenek. Emiatt az elektronok útvonala könnyen kijelölhető, olcsón, történetesen rézhuzalból, megépíthető;

— *stabil*, nem nyelődik el, mint akár a *fény*.

Amikor gépeitenni kívánunk valamit, nagyon fontos, hogy olyan munkaeszközünk legyen, amivel számtalan műveletet — például rengeteg gondolatfeljegyzést — végezhethetünk el igen gyorsan, igen kis energiával és teljesítménnyel. A legegyszerűbb, naponta gyakorolt formája a feljegyzésnek, hogy ceruzával strigulákat húzunk egy papírlapra. Az elektronikában ennek az felelhet meg, hogy egy mikroszkópikus méretű elektrontrájt (kondenzátort) feltöltünk, ma legalább mintegy ezer elektronnal. Ha a tartály megtelet — ott a strigula, ha a kondenzátort kislüktük: kiradirozzuk a strigulát. Elektronokkal dolgozva mindezt megtehetjük — ha úri kedvünk úgy kívánja — akár százmilliószor is másodpercenként.

Egy bitnyi információ (hogy ez mit jelent, arról a Magazin 1987/10. számában már írtunk) feljegyzése elektronokkal ma mintegy 10^{-16} joule energiát igényel, és még nem tartunk a fejlődési folyamat végén; az elméleti határ mai ismereteink szerint 10^{-21} joule nagyságrendnél van. Hogy ez technikaileg mennyire közelíthető meg, az a jövő kérdése. Mindenesetre úgy látszik, van még jócskán lehetőség a fejlődésre.

Elektronika — erősáramú technika

Nem minden szerkezetet nevezünk *elektronikusnak*, ami elektronok tulajdonságain alapuló hatások kihasználásával működik. A határ az elektronikus és az ún. erősáramú szerkezetek között elmosódó, inkább gyakorlati, mint elméleti. Egy nagy teljesítményű villamos motort vagy áramfejlesztő generátort senkinek sem jutna eszébe *elektronikusnak* nevezni, holott működésükben kétségtelenül a mozgó elektronok játszzák a kulcsszerepet. Hasonlóképpen nem szokták elektronikusnak nevezni azokat a szerkezeteket, melyek *nagyobb teljesítmények* átalakítását szolgálják vagy melyekben *mozgó mechanika* alkatrészek jelentős szerepet kapnak. Nem szokták például elektronikus szerkezeteknek tekinteni a mozgó alkatrészeket nélkülöző félvezető alapú átalakítókat sem, melyeket többek között számítástechnikai szünetmentes áramellátására alkalmaznak. (Ezek egy akkumulátortelep egyenfeszültségűt alakítják át 220 V-os hálózati feszültséggé nagy teljesítményű félvezető segítségével.) De — hogy egy ellenpéldát is idézzünk — már a nagy teljesítményű számítógépekbe beépített, ugyancsak nagy teljesítményű, ún. *tápegységeket*, melyek a hálózati feszültséget alakítják át nagy teljesítményű félvezető áramkörök segítségével a gépbe beépített integrált áramkörök tömegének energiaellátásához szükséges (ma tipikusan 5 V körüli) egyenfeszültséggé, azokat már elektronikus eszközöknek tekintik az utóbbiakban.

Szerepe van az elektronika és az erősáramú technika elhatárolásában az egyetem, a főiskola stb. oktatásnak és az ipar világszerte nagyjából egyöntetű, de sokszor esetleges szakosodásának is: külön képeznek például szakembereket „erősáramú” és külön „gyengeáramú” technikára; általában más profilú vállalatok géptanuk áramátalakítókat és mások számítógépekbe beépítésre szánt tápegységeket stb.

A furcsa határesetek számbavétele helyett ma elfogadható egyszerűsítéssel azt mondhatjuk: *elektronikus szerkezeteknek ma és még jó ideig azokat nevezhetjük, melyek működésük lényegét tekintve tipikusan félvezető alapú, integrált áramkörökből épülnek fel.* — A továbbiakban eszmefuttatásunkat a digitális elektronikára korlátozzuk. Hogy ez mit jelent, arról lapunk 1987/9. számában írtunk.

Integrált áramkörök — elektronikus szerkezetek

A *digitális*, félvezető alapú integrált áramkörök nagyon sok, mikroszkópikus méretű „elektrontrájt” — kondenzátort — és az elektronok mozgását szabályozó, igen gyorsan vezérelhető (nyitható és csukható) kapcsolót, tranzisztort tartalmaznak. Ma néhány négyzetmilliméternyi területre, a chipen vagy morzán néhány száz ezer vagy akár néhány millió tartály, kapcsoló helyezhető el úgy, hogy a tartályok és a kapcsolók akár a másodperc százmilliomod része alatt nyithatók-csukhatók, feltölthetők-kislüktethetők. Egy-egy ilyen integrált áramkör üzembentartása ma, a „szívárgások”, „sűrűlódások”, a csatlakoztatások stb. miatt, tízeduzz nagyságrendű teljesítményt igényel. (Ha egymillió mikroszkópikus méretű kondenzátort töltögetnénk és sűrűtögetnénk ki másodpercenként egymilliószor, és kislüktetünk megannyak az energiát elveszteni, akkor ehhez csupán 10^{-4} watt teljesítményre lenne szükség!)

A digitális elektronikus eszközök tervezése — nem túl nagy egyszerűséggel — nem más, mint sok millió tartály és vezérelhető kapcsoló célszerű összeköttetésének meghatározása. A mai átlagos feladataink megoldásához szükséges darabszámok azonban akkorrak, hogy nagyon szakszerű és gondos *struktúrálás* és az ennek megfelelő munkaszervezés-munkamegosztás bevezetése nélkül az összeköttetések tervezése megoldhatatlan lenne.

Az elektronikában belül szervezetileg is (külön vállalatok, külön szakközvetítés formájában stb.) elkülönülnek azok, akik a félvezető lapkán belüli összeköttetésekkel — mikroelektronikával — foglalkoznak, azoktól, akik *kéz*: integrált áramkörökből építkeznek. Ennélfogva elkülönülnek tehát a szerelőipartól, az eszközterveztől, a berendezésgyártástól és általában a számítástechnikától stb.

A félvezető alapú integrált áramkörökből felépülő elektronikus szerkezetek (hardverek) készülhetnek valamilyen meghatározott célra és lehetnek „univerzálisak”. A potenciálisan „univerzális” számítógépek egy-egy konkrét feladat elvégzésére programozhatók (szoftver). A programok maguk is gyakran olyan sok lépésből — utasításból, parancsból — tevődnek össze (megjegyzem, egy átlagos könyv hozzávetőlegesen 20 ezer sorból áll; vannak 10^5 , 10^6 utasításból álló, sőt még ennél nagyobb programok is!), hogy komoly elméleti alapokon nyugvó struktúrálás (mint a nyelvi, fordítói, operációs, „run-time”, keret-stb. rendszerek) nélkül áttekinthetetlen lennének. A mindezzel való foglalkozásnak az a jelentősége, hogy az ember gondolkodás, a megismerés, az absztrakció, a tanulás, az intelligencia, a nyelv új dimenzióit felfedezze, tárja fel. Mindez az elektronikus elven működő szerkezetekhez tapad, mindezt az elektronizáció részeként tartjuk számon. Valószínű, hogy épp ez az „*tapadékok*” az, ami az elektronizációval kapcsolatban hatásaiban és feltételeiben leginkább tanulmányozásra érdemes.

Az elektronizáltság, mint a fejlettség fokmérője?

Napjainkban az országok egymás közötti nem hivatalos versenyében a sorrendet az elektronizáltság fokának megfelelően szokták meghatározni. Úgy tűnik, az elektronizáció nem pusztán technika-technológiai, hanem fontos társadalmi és politikai kérdéssé is vált. Emiatt gyakran merül fel a kérdés, mérhető-e egy-egy országban az *elektronizáltság* foka.

Ha elfogadjuk a korábbi gondolatmenetet, akkor — mint előbb már megfogalmaztuk — célszerű kis egyszerűsítéssel azt mondhatjuk, hogy minden *elektronizáció*, ami alapvetően, lényegét tekintve félvezető alapú in-

és most először az elektronika területére inváziójuk önöket, majd hírt adunk egy amerikai felsőoktatással kapcsolatban könyvről. Mindenki tudja, hogy mai számítógépeink elektronikusak, de vajon hányan tudják azt is, mit jelent ez?

tegrált áramkörök beépítéséből áll, továbbá olyan eszközök felhasználásából, amikben ilyenek vannak.

Ugy tünhet, ezek után a „verseny” kiértékeléséhez nem kellene mást csinálni, mint azt összeszámlálni — valamilyen egyszerű súlyozással —, hogy a vizsgált ország, népesség egy-egy tagjára hány millió elektron-, tartály- vagy kapcsoló* jut. Minél több, annál elektronizáltabb a vizsgált ország, a vizsgált népesség. A súlyozás alapja lehetne például a maximális működési sebesség, a fajlagos energiaszükséglet, a megbízhatóság, a környezetlélősség stb. Nagybő súlyt kapnának esetleg a gyorsabb, magasabb hőmérsékleten működésre képes, megbízhatóbb integrált áramkörök. Feltehetőleg tehát, hogy az integrált áramkörök száma általában meghatározza a felhasználóknak szükséges szoftver mennyiségét is.

Mindez elméletileg szép, csakohy amíg a hazánkban gyártott integrált áramköröket meg az importáltakat számításba lehet venni, ki fogja összeszámlálni a legkülönbözőbb importált eszközökre, szerelvényekre — a hazai gyártók által megvásárolt ún. „produktív importba” — beépítettek is? A mi külgazdasági nyitottságunknál ez nem elhanyagolható mennyiség. Továbbá: igaz-e, ha például egy gépkocsiba beépített mechanikus szerkezetű órát felvált egy elektronikus szerkezetű óra, akkor ez feltétlenül a fejlettség, előrehaladottság jelzője??

„Mindazon „gyanús” dolog közül, ami a világot arculatát valószínűleg (talán?) meg fogja változtatni — ha egyáltalán ki lehet ragadni egyet vagy néhányat a sok közül —, az nem általában az elektronizálás lesz, hanem annak egy része, következménye, a szellemi munka át- vagy felértékelődése:

- a szellemi munka gépésíthető (rutin-) részenek gépésítése és ezáltal a munka humanizálása (?);
- a gondolkodás új területeinek meghódítása (?);
- az intelligencia, a tanulás, a nyelvezet stb. mélyebb megértése.

Valószínű, hogy egy-egy ország elektronizáltsága szoros összefüggésben van az adott országban működő számítógépek számával, illetőleg e számítógépek *összeteljesítményével*. Ha ez így lenne, akkor egy-egy ország, népesség elektronizáltsági fokának jellemzésére használhatnánk az egy főre jutó számítástechnikai teljesítmény mérőszámát. A különböző számítógépek teljesítményének közös nevezőre hozása — egy vagy néhány mérőszámmal való jellemzése — sem egyszerű feladat. Ebből a szempontból érdekes például a különféle, például a COCOM tilalmi listának alkalmazott számítás.

Most pedig következnek a híradás Allan Bloom „The Closing of the American Mind” (Simon and Schuster, 1987. 392 oldal) könyvéről.

Az amerikai gondolkodás beszűküléséről

szól Bloom professzor könyve, amihez a neves amerikai író, Saul Bellow írt előszót. Arról, hogy szerinte az amerikai felsőoktatás hogyan értette félre a demokráciát és miként szegényítette el a mai diákok lelkivilágát. A szerző, aki maga is tanított patinás, ún. ivy league egyetemeken, Saul Bellow szerint kiváló könyvet írt már Shakespeare politikájáról, lefordított a Platon „Közértés”-át, Rousseau „Emile”-jét. Ilyen ajánlólevéllel nehez lesz egy közlegyintéseltélinzi súlyos, kritikai mondanivalóját. Olyan tekintélyek szólnak mellette, mint Szókratész, Platon, Arisztotelész, Machiavelli, Rousseau és Kant.

A könyv három részre tagolódik: a *diákokról*, a *nihilizmusról* és az *amerikai életszilárd*, végül az *egyetemről* szólóra.

Rovatunk gyakorlatához hően, általában áttekinthető helyett vegyünk inkább nagyobb alá egy kisebb, jellemző részletet. A diákokról szóló első rész négy fejezetre, a negyedik pedig több alfejezetre oszlik: Tiszta lappal; Könyvek; Zene; Kapcsolatok; Egocentrikusság; Egyenlőség; Fajji kérdések; Nemek közötti kapcsolat. Állandó kötődések hiánya, Válás, Szerelmek, Erosz. Nézzük meg közelebbről a nemek közötti kapcsolatokról szóló alfejezetezt.

Itt a szerző arról ír, hogy a közhiedelemmel ellentétben Amerika nem valamiféle meglepetlen, antitelmlektuális ország, ahol az elméletet legfeljebb csak arra kellene, hogy igazolják a célokhoz szükséges cselekedeteket. Epen ellenkezőleg: Amerika olyan színpad, ahol a tragédiák és komédiák főszerepét mindig is a nagy elméletet játszótták. Amerika a szerző szerint filozófusok és tanítványok országa. Minden cselekedetet a *szabadság* és *egyenlőség* elve vezérel.

Ez a két vezérel most fellép a legmagasabb magánélet színpadára, hogy

eljátssa utolsó főszerepei egyikét. A nemek kapcsolata — és minden, ami még hozzá tartozik: a szerelmek, a házasság, a család — most ebben a végjátékban egy nagy nemzeti elgondolás, terv tárgya lett. De a természet itt úgy látszik, ellenáll az emberi törekvésnek, ellenáll annak, hogy az ember ide is kiterjessze a mindenhátot eszmék hatalmát.

A nemek közötti kapcsolatokat két hullámban érte kihívás jellegű változás. Az első *szexuális forradalomnak* nevezhetjük, a második a *feminizmus*. Az elsőnek a *szabadság*, a másodiknak az *egyenlőség* volt a zászlajára írva. A kettő egy darabig jól megvolt egymással, de már Tocquville is megmondta: a szabadság és egyenlőség nehezen egyeztethető össze. A feministák harcot kezdtek a pornográfia ellen. A pornográfia védelmezői a szőlőszabadság páncélzatát lötték fel. A „forradalmi” feministák így furcsa mód hírtelen egy táborba kerültek a közönség morált, a hagyományos nemi szerepeket védő *konzervatívokkal*, akik tabunak tekintettek volna minden kapcsolatot aközött, amit az emberek olvasnak vagy képeken látnak és aközött, amit egymással csinálnak. És e küzdelem hátterében ott álldogáltak, kezük aggodalmasan tördelve a *liberalizmus*ok, akik mindkét félnek szerettek volna igazat adni, de nem tudtak.

A szexuális kapcsolatok felszabadításáért indított mozgalom az ösztönöket fel akarta szabadítani a puritán bibliai örökség, az erendendő bűn eszméjének terhe alól. A hatvanas évek fokozatosan kitapogatták volna a szókimondás határait, de azok végül is egyszerűen szétolvadtak anélkül, hogy nagy feltűnést támadt volna ebből. A szülők és tanárok azon is elég hamar túltették magukat, hogy a diákok kezdtek együtt elni és hálni. Az erkölcsi gatlakosok, a betegségtektől való félelem, a terheség kockázata, megfelelő hely hiánya — szóval mindaz, ami korábban ennek útjában állt — egyszerűen csak eltűnt. A diákok — különösen a lányok — egyszerre csak elkezdtek nem szégyellni azt, hogy kimutassák szexuális érdeklődésüket, sikereiket. Az együttlások, mely a húszas években még veszélyesnek, a harmincas-negyvenes években még jó esetben is legfeljebb bohémkodásnak minősült, egyszerre olyan természetessé kezdett válni, mint valami lánycsérkés-csapatheli tagság. Azért kell azt mondani, hogy „különösen a lányoknál”, mert a tradicionális gondolkodás (hit?) szerint a fiúk mindig kézen lettek volna habozás nélkül erre, és e szereszotás a fiatal nők mindkét várta volna el a szerelmes tartózkodást. Az első hullámban tehát ki kellett kapcsolni ezt a tradicionális női szemérmeket. Mivel ezt a szemérmeket pusztán valamiféle konvencionális, szokásának fogták fel, nem jelentett problémát a leküzdése. Ez az első hullám, a szabadság csak hangsúlyosabbá tette a nemek közötti különbségeket: mind a férfiak, mind a nők hangsúlyosabban férfiak és nők szettek volna lenni.

Ez a nagy felszabadulás azonnali boldogságot ígért, felszabadított érzékeink ezereves sötét elnyomása alól, feloldódást valamiféle nagy, folytonos bacchanáliában. De a ketreche zárt fenevadról hamarosan kiderült, hogy nem orozslán, hanem nagyon is házasított macska eszupán. Ami a nagy felszabadítás során kiderült, az az, hogy az egész szenvedély nem is olyan veszedelmes, jobban zóld utat engedni neki, mint a lázadás elnyomni. Bloom professzor egyszer megkérdezte diákjait: hogy lehet az, hogy nemrég még a szülők rászóltak a kimiradós lányukra: „Nehogy nekem még egyszer leoltsd a bejárati lámpáit”, most meg akkor sem szólnak, ha a fiúk barátjává alszik. Erre azt a feleletet kapta egy csinos, nagyon normális, fiatal aszszonykától, hogy: „Nagy dolog!” És ez az, ez a szenvedélynélküliség a legjellemzőbb eredménye az első „forradalmi hullámnak”.

A második hullám, a feminizmus, a felszabadítás köntösbéiben érkezett. Sokkal inkább a természeti kötöttségektől, mint a megszokástól való felszabadításában. Ezért komorabb, erotikamless, valamiféle elvontabb társadalmi reformtervezést volt. Kevésbé állt szemben a fennálló törvényekkel. De, hogy mi a végső cél — mint Freud is utalt rá —, nem volt olyan világos.

A szóhasználat eltörodött az „*eljünk természetesen*”-től (amellyel legtöbbször nagyon is meghatározott testi funkciókra utaltak) olyasmire, mint „*legyelj saját magad*”, „*teljesítsd ki magad*”, „*magad határoz meg prioritásaidat*”, „*alaktisd ki magad saját életiltusodat*” stb. A nőmozgalom Bloom professzor könyve szerint nem az emberi természetre alapozott. Az alfejezet a továbbiakban elsősorban a feminista eszmék hatását boncolgatja igen tanulságos módon.

E rövid izellítovel reméljük, sikerült olvasóink érdeklődését felkelteni a könyv iránt.



Kiválóan alkalmazható a
Grabiol AV 04
antisztatikus padló ott, ahol az
elektrosztatikus
feltöltődésből eredő problémák
fokozottan jelentkeznek:

tűz- és robbanásveszélyes üzemekben, laboratóriumokban, számítógéptermekekben, kapcsoló, vezérlőhelyiségekben, telefonközpontokban.

A Grabiol AV 04

speciális tömör PVC-padló

a feltöltődés szempontjából vezető, a kisülés szempontjából szigetelő, elektrosztatikai tulajdonságai mind a vezető-, mind a szigetelőanyagokénál kedvezőbbek.

Fajlagos térfogati ellenállása 10^3 – 10^6 ohm/m³,
500 V mérőfeszültségen mérve.

A Grabiol AV 04 2 mm vastag

2 m széles

11 m-es tekercsekben kerül forgalomba.

Kiegyenlített betonaljzatra hegesztve fektethető, de jó álpadlórendszer (pl. Betonyplap) burkolására is.

A gyártó Graboplast

— vállalja a padló speciális rézháló nélküli fektetését, mely gazdaságosabb és jobb a hagyományos módszernél,

— a kivitelezéshez szükséges anyagokat adja,

— a mérési jegyzőkönyvvel igazolja, hogy a kész padozat az MSZ 16040/1–73 és a 16041 szabványnak megfelel.

Bővebb felvilágosítással készséggel szolgálnak a gyár szakemberei.

Megrendelhető a Belföldi Értékesítési Osztályon.

Graboplast

Győri Pamutszövő és Műbörgyár

9023 Győr, Fehérvári út 16.

Telefon: 96/14-155

Telex: 24-276. Telefax: 18-306

PROGRAMTERMÉK

Zeneközselben – programmal!

Volt egy korszak, amikor a számítógépes zeneszerzést gunyiratinkban emlegették, mint a rothadó kapitalizmus újabb, velejéig romlott ötletét. Azóta – a személyi számítógép eljövetelel – óriási néptömegek kaptak lehetőséget az önálló számítógépes zenészenre – még a keleti blokkban is! –, és személyesen győzdhettek meg a dolog értelméről vagy értelmetlenségéről. Azoknak a szerencséseknak, akik tehettek, nagy kedvük telik az akármilyen tárgyú zenei lehetőségekben is.

A Novotrade-től kapott egyik legújabb programcsomagban két zenepedagógus próbálkozik a zenetanítás számítógépes támogatásával. A gépi lehetőségek előnyeit és korlátait figyelembe véve, nem határozták el az énektanárokat – és önmagukat – lecsérésését, sőt kompromisszumot kellett kötniük a géppel támogatható és az iskolai tantervben előírt ismeretek körére vonatkozóan is. Az elkészült oktatási program-csomag így „három és feledek” generációsnak tekinthető, amennyiben bizonyos mértékig támaszkodik az előírt tematikára, de ahol csak lehet, elrugaszodik attól, és a számítógépes lehetőségeknek megfelelően ástrukturálja az átadandó ismereteket.

A program jellemzőit a szokásos, összefoglaló adatok táblázata mutatja.

A programcsomag kezelése nagyon egyszerű. A kazettákról a betöltés elég hosszú időt vesz igénybe a turbóbetöltő ellenére is, különösen, ha a szalag végén lévő programot kell betölteni. A programban gyermekem segített egy jó trükkrel. Elmondom, hátha más is tudja hasznosítani. A kazettákról a turbóbetöltő már elindult, akkor leállítjuk a kazettaolvasót. Kivesszük belőle a kazettát, és egy normál magnó gyorsan előrecsévéljük a szalagot a betölteni kívánt program előtti hangtalan szalagszakaszra. Ha szerencsénk van, a kazettát visszatehetjük a magnóba, és szinte azonnal töltődni kezd a keresett program (ha füllel sikerült általánai az üres szakaszt!). A programcsomagot hosszabb kazettákra írták fel, hiszen azokon három-öt

program is található. Az igazság az, hogy ezzel a programcsomaggal tantermi órákon csak mágneslemezzel lehetne jól érvényesülni. A lemezegység azonban a C Plus/4-ek környékén fehér holló.

A program dokumentációja elég lakonikus, de sajnos ebben nemzetközi normákhoz igazodik. A programkazettákhoz általában csak anynyi leírás mellékelnek, ami a csomagolóanyagban elfér. Ugyanakkor elmondható, hogy a programok öndokumentált jellegűek. Döntő többségében tudni lehet, hogy egy adott képen éppen mit lehet csinálni és mit nem. A borítón a szöveget sajnós iróppal írták, ezért a hosszú ú és i betű helyett azok rövid párja szerepel.

A Ritmus I. kazetta a démon kívül az általános iskolai I–IV., a Ritmus II. kazetta az V–VII. osztályos ritmusanyagot tartalmazza. A Dallam I–II. nem kötődik közvetlenül évfolyamtantervekhez, hanem olyan zenei alapelvekhez, amelyek keresztül-kasul átszövik az általános iskolai tananyagot.

A szerzők úgyesen és játékosan oldották meg a ritmusok gyakoroltatását. A szóközbilentyű a C Plus/4-en talán elég stabil erre a célra. A ritmusokat meghallgatás után utánozni kell, meghozzá igen kényes pontossággal. Ez a pontossági igény már kezdő nehézségi foknál is igen erős. A Novotrade-nem miatt sokan panaszkodtak, hogy nem jó a program. Ez nem így van. Követelményeket támaszt. A nehézségi fok egyébként beállítható, tehát a gyengébbek ritmusérzéke javítható, a tehetségesebbek pedig tovább fokozható. Ezt nagyon helyes alapelvnek tekintem.

A programcsomagban sikerült elég szép képi megjelenítést alkalmazni. A kottarajzok igazán szépek, világosak és érthetőek. A dallamgyakorlatokban egy helyen azonban nehezen jöttem rá, hogy az egynegyedes hangnak két kottafaj hely kell a kottában. Mindenesetre úgyesen szervezték a kották bekeresését, a „zenegrafikus inputot”. A menüvezérlés egyszerű,

de a célnak megfelelő. Nem kell, hogy zenepedagógusok ilyen témákban túl nagy erőfeszítést igénylő megoldást hozzanak.

Jó poén, hogy a moilhangos-gyakorlatok közül a 9 éppen Beethoven IX. szimfóniájából az örömdát idézi. Emellett úgy érzem, hogy a szerzők a Bartók–Kodály hagyományok megőrzésére törekedtek, amennyiben a dallamanyag főként magyar népdalokra épül.

A gyakorlatok némelyike egy becsapós kérdés/válasz üzenettel lepett meg: folytathatom (I/n), de a nem választ ignorálta. Zárt hurokba kényszerít, azaz nem engedi, hogy kiszálljak a feladat megoldása nélkül, ami alapjában véve nem rossz elv. Ilyenkor én csak „I” választ írnék a programomban. A programcsomag drill módszerei egyébként jó eredményeket ígérnek, ha a tanuló elég szorgalmas. Én tudom hinni, hogy a programmal olyan tanuló – vagy akár felnőt! – ritmusérzéke és zenei készsége is hatékonyan javítható, aki tipikus zenei analfabetának ismeri magát. Már csak az a lehetőség is indukál, hogy a tanórai stressz nélkül, nyugodt környezetben is lehet tanulni a programcsomaggal, és a haladási sebességet is ki-kimagához igazíthatja. Minimális szorgalom persze nélkülözhetetlen a sikerhez. Korszerű elemnek tartom azt a megoldást, hogy a program lehetőleg segíteni akar. Például bizonyos szituációban a tanuló kézbizonytalansága ellenére is addig tartja ki a hangot, amíg kellene.

A tartalom elemzésére nem vállalkozhatok, jóllehet énekkarokban énekeltem, de mikor volt az már. A rendelkezésre álló rövid idő alatt elég kevés modulba sikerült beletekinteni. Maguk a szerzők is kapacitásnak arra, hogy a számítógépes oktatás sohasem helyettesítheti a tanárt, hiszen a gép az éneklést már nem tudja visszaellenőrizni, mert nem hall (egyelőre!). A négykazettányi méret mégis azt sugallja, hogy a szakembeli szerzők az elérhető maximumra törekedtek.

Én sem dicsekszem azzal, hogy már a programcsomag legelső modulja is olyan nehéznek bizonyultak számomra, miszerint gyermekeim ügyességét kellett igénybe vennem, ha túl akartam jutni legalább egy-egy típusfeladaton. Mégis megerősítem, hogy ez csak azt mutatja, a programcsomag – nagyon helyesen – erőfeszítésre serkentő követelményeket állít a tanuló elé.

A program az általános iskolai munkát és bármely korosztály önképzését támogathatja, ami önmagában is igen jelentős teljesítmény.

Zsadányi Pál

ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	Novotrade Rt.
Terméknév:	Ének-zene Ritmus I-II. Dallam I-II.
Szerzők:	Brusznay Margit Polgár László
Géptípus:	C Plus/4
Hordozó:	négy kazetta
Dokumentáció:	két-két oldal a kazettaborítókön
Ár:	Ritmus I. 798 Ft Ritmus II. 798 Ft Dallam I. 610 Ft Dallam II. 498 Ft

MINŐSÍTŐ ADATOK

Kezelhetőség:	kiváló
Teljeség:	kiváló
Dokumentáltság:	jó
Használhatóság:	kiváló
Ár/teljesítmény:	jó
Összbenyomás:	kiváló

A figuratípusok jutalompontjai

Az értékelőfüggvény főbb összetevőit az eddigiekben már részletesen tárgyaltuk, sőt Dap Hartman ANALYSE programjának segítségével kiemelt nyolcszázharminckét mesterjátsma elemzésének eredményét is bemutattuk. A sakkprogramok fejlődése azonban nem áll meg, a sakkprogramozók mindig újabb szempontok figyelembevételével próbálnak javítani programjukon.

A híres Chess 4.5 sakkprogram értékelőfüggvénye az általunk is már jól ismert komponensek pontjain kívül minden figurának külön bónuszot adott, amelyek nagyságát számos tényező határozta meg. Ezek a bónuszok nagymértékben növelték a program játékeréjét. Ezt az bizonyítja, hogy 1970-ben, amikor a Northwestern Egyetem diákjai — David Slate, Larry Atkin és Keit Gorlan — elkészültek sakkprogramjukkal, megnyerték vele Észak-Amerika első bajnokságát. Az ezt követő tíz évben is csak kétszer adták át első helyüket, először 1974-ben a kanadai Ribbit-programnak, majd 1978-ban a Belle-nek, amely később ugyanígy zsinórban nyerte meg Észak-Amerika bajnokságait.

Dap Hartman, akinek tanulmányából már többször idéztünk, feldolgozta a Chess 4.5 programban használt bónuszokat is. Ellenőrizte, hogy a figurák elhelyezkedésére adott jutalompontok milyen mértékben reálisak. Ehhez 832 mesterjátszmát használt fel, amelyekben minden lépés megtétele után kiszámította a jutalompontokat. Ezután kiszámította az összes játszámra vonatkoztatva az azonos lépésszámhoz tartozó bónuszok átlagát mindkét fél számára, amit a féllépésszám függvényében ábrázolt külön a nyertes és a vesztes játékosnál (a vezér bónuszának függvényét az 1. ábrán láthatjuk).

Nézzük, hogyan alakulnak a bónuszok az egyes figuratípusokkal. A vezérre nagyon egyszerű a jutalompont kiszámítása:

$$F_q = 0,8 \cdot A - 8 \cdot D, \text{ ahol}$$

A: a vezér által megtámadott olyan mezőknek a száma, amelyeket nem támad ellenséges gyalog, futó vagy huszár.

D: a vezér és az ellenséges király sorkülönbségének és oszlopkülönbségének a minimuma.

Dap Hartman ANALYSE nevű programja enél is egyszerűbben számolt a nagymesterjátszmák kiértékelésénél, ugyanis a 0,8-as szor-

zót elhagyta, ezért a függvény a következőképpen módosult:

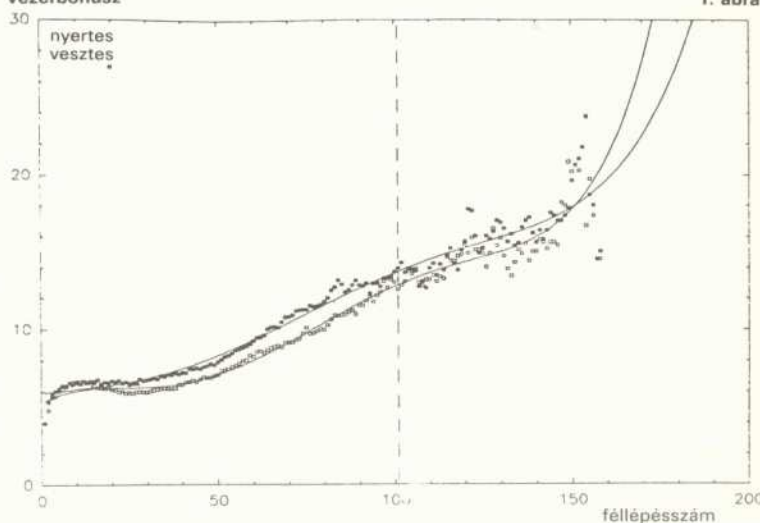
$$F_q = A - D, \text{ ahol}$$

A és D jelentése azonos maradt az eredeti függvényével. Ezzel a módosított függvénnyel számítottuk ki a 2. ábrán látható hadállás vezéreinek bónuszát, amely világosnál és sötétnél is kilenc. Világos vezér által megtámadott mezők: e2, e1, d1, c1, c2, b2, c3, b4, a5, d3, d4, d5, d6, d7, d8, e3, f4, g5, h6. Ezek közül figyelmen kívül kell hagyni azokat, amelyeket az ellenfél gyalogjával, futójával vagy huszárjával támad.

Ezek a következők: b4, a5, d4, d5, d7, d8, h6. Tehát összesen tizenkét olyan mező van, amely megfelel az elvárásoknak, ezért erre 12 pontot adunk. Ebből viszont le kell vonni a vezér és az ellenséges király oszlopkülönbségének és sorkülönbségének minimumát. Világos vezér d2-n, sötét király g8-on helyezkedik el, ezért az oszlopkülönbségük a kisebb, mégpedig három. Tehát a végső pontszám a 12-3, azaz 9. Ha a sötét vezért vizsgáljuk hasonló módon, szintén kilencet kapunk.

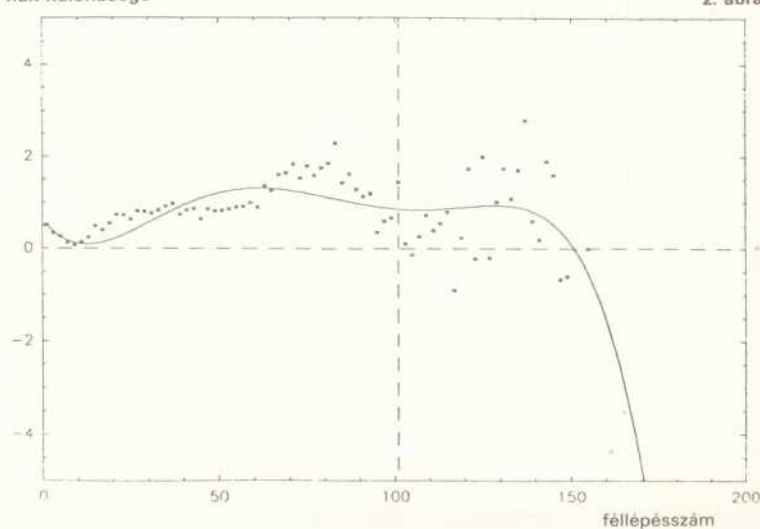
Kovács P. Attila

Vezérbónusz



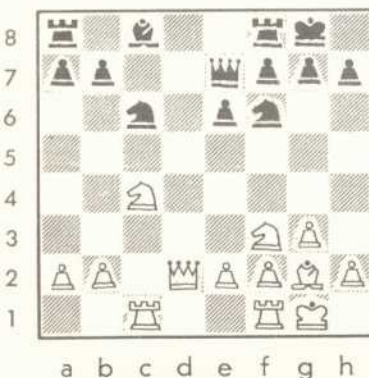
1. ábra

A nyertes és a vesztes vezérbónuszának különbsége



2. ábra

Kaszparov—Petroszjan, Bugojno 1982.
A 24. féllépés utáni hadállás



Könnyen választhat a SZÁMALK MENÜ-jéből

A SZÁMALK értesíti az érdeklődőket, hogy a MENE-DZSER-üzletága, a SZÁMALK—MENÜ az alábbi termékeket ajánlja.

A MENÜ nagy- és demigrosz-kereskedelmi tevékenységet folytat, főként a professzionális számítástechnika területén (8 bit felett).

Megrendeléseket veszünk fel:

- a legújabb, komplett külföldi szoftverekre
- hardverkiegészítőkre
- hardverre
- CAD-rendszerekre
- tanácsadásra
- szervezésre, programozásra

A SZÁMALK—MENÜ a MICROSOFT magyarországi disztribútora: minden MICROSOFT szoftvert és Update-et 3 héten belül szállít írásos megrendelésre. Ajánlatkérésre postafordultával válaszolunk.

Termelőktől és más forgalmazóktól továbbforgalmazásra átveszünk jól dokumentált szoftvereket. Szükség esetén megszerzzük az adaptálást.

MAGYAR TERMÉKEINK

SEGÍT szoftver

IBM PC/XT, AT kompatibilis gépekre. Tárrezidens DOS HELP. Minden felhasználónak magyar nyelvű tájékoztatást ad KÉPERNYÓN a DOS használata közben. Gyűjtött és szerkeszthető DOS-utasítások. A felhasználó bővítheti, programozhatja a saját SEGÍT-jét. Ára: 4500 Ft.

SZÜNTEMENTES TÁPEGYSÉG 600 VA-es (AKKU-párolás elleni védettséggel). 3 óras áramkimaradás mellett is tovább dolgozhat IBM PC/XT, AT kompatibilis gépen. A BITBOLT-ban megtekinthető: 1 db AT, 1 db XT és FX 1000. Ára: 120 ezer Ft.

SECRET

Védi dokumentumait IBM PC/86-286-386 komputerén az illetéktelenektől. A SECRET adat- és programvédelmi rendszer, szoftver- és hardvereszközökkel.

Kit, mit véd?

- a felhasználót
- a directoryt
- a programfájlt
- az adatfájlt

A MENÜ iroda rendelési címe:

1123 Budapest, Kapitány u. 6. /1/.

Telefonszám: 565-419

A BITBOLT címe: 1136 Raoul Wallenberg u. 5.

Telefonszám: 110-983

Mivel?

- passworddel
- kóddal
- hardverkártyával (opció)
- EPROM-kulccsal (opció)

Hogyan?

- a dekódolhatatlan password-kód kombinációjával
- hozzáférés előtt illetékesnek nyit, utána rögtön zár a védelem (a sebességsökkenés maximum 10%)
- hardveres opció esetén a floppy-boot letiltásával
- törölhetetlen fájlokkal
- átnevezhetetlen fájlokkal
- lemasolt védett fájlok illetéktelenek által nem használhatók

Kitől?

- Más illetéktelen felhasználótól.
- Szükség esetén a systemmenedzsertől is.

Hol?

- Egy gépen.
- Hálózatba épített több gépen is.

Meghibásodáskor:

- A systemmenedzser hardverűton felélesztheti a rendszert. Kérje árajánlatunkat szoftverre, kártyára, kulcsra!

ELTGUARD szoftver: programvédelmi rendszer szoftverűton. Hierarchikus programvédelem. Korlátozott felhasználói hozzáférés biztosítása programrendszeren belül. Menüvezérlés. Nem csak egy gépen használható. Több mint két éve feltörhetetlen! Ára: 120 ezer Ft. (Adatvédelmi kiegészítő elkészülte után térítésmentesen a korábbi vevőknek.)

MEDANIN

Orvosok, klinikusok és farmakológusok gyakorlati munkáját segíti az adattárolásban és analízisnél. Válaszok biológiai, diagnosztikus és terápiás kérdésekben. MEDIKUSOK ANALIZIS és INFORMÁCIÓS szoftver a MEDANIN. Készítette prof. dr. Berentei és prof. dr. Kékesi. Ára: 100 ezer Ft.

KIR

Winchesterén és floppyjain tárolt fájlokból directory csoportosításban saját katalógusfájlt készíthet. Megjegyzésekkel elláthatja, csoportosíthatja, dBASE típusú katalógusfájlt tovább feldolgozhatja. Ára: 15 ezer Ft.

Rendelésre — rövid határidővel — szállítjuk a fenti szoftvereket, illetve hardverkiegészítőket. Felhívjuk szíves figyelmüket, hogy megadott áraink nettó — áfa nélküli — árak.

Kérésre árkatalógust küldünk több mint 1500 legális szoftverre. Szükség esetén tanácsadás.

**Kérjen levélben ajánlatot,
árkatalógust!**

Válaszunk után írásban rendeljen!

Rövid szállítási határidő

Könnyen választhat a SZÁMALK MENÜ-jéből!

Tisztelettel a
SZÁMALK—MENÜ
ÜZLETÁG



Marchant, J. J.
Számítógépes műszaki tervezés.
Az előkészítés gyakorlata
 (Budapest, 1988.
Műszaki Könyvkiadó, 144 oldal.
Ára: 85,— Ft.)

A digitális számítógépek, valamint grafikus be- és kimeneti eszközök új korszakot nyitottak a fizikailag nem létező objektumok szemléltetésében. Ilyen objektumok a fejlesztési folyamat kezdeti szakaszában még csak a tervezőmérnök fejében létező műszaki alkotások, amelyeknek megvalósítási módjait papírra írják. A digitális számítógépes rendszerek alkalmazásával a mérnöki műszaki megoldásokat számítógépes belső ábrázolási modellként lehet szemléltetni. A tervező és a gép közötti kommunikáció eszközei: a billentyűzetek, a digitizáló táblák, a fénylók, a képernyős megjelenítő eszközök és a számítógépes rajzolóberendezések. A digitális számítógépes eszközöket alkalmazó termékfejlesztési folyamat egészét nevezik számítógéppel segített tervezésnek (CAD = Computer Aided Design).

A CAD alkalmazásának előnyei közé tartozik, hogy a tervezői elképzelések szemléltetéséhez nemcsak a klasszikus grafikai alapelemek — a pontot és a vonalat — lehet felhasználni, hanem felületeket, térfogatokat vagy akár alkatrészeket és szerkezeti egységeket is. A hatékony számítási eljárások — például a véges elemes módszer — lehetővé teszik bonyolult rendszerek viselkedésének számszerű vizsgálatát. Az adatbázisrendszerek megkönnyítik a meglévő megoldások visszakeresését.

A CAD rendszer fejlesztőjének ki kell választania a piacon található nagyszámú rendszerből a számára legmegfelelőbbet, amelynek egyúttal illeszkednie kell a megvalósítandó, számítógéppel integrált gyártórendszeréhez.

Ez a könyv a viszonylag új keletű, CAD-höz kötődő ismeretek hiánypótló mű, egyúttal tanácsokkal segíti a saját CAD-alkalmazások kialakítását.

**Kernighan, B. W. — Pike, R.:
 A UNIX operációs rendszer**
 (Budapest, 1988.
Műszaki Könyvkiadó, 362 oldal.
Ára: 298,— Ft.)

A UNIX a legszűkebb értelemben egy időosztásos operációs rendszer magja, olyan program, amely vezérli a számítógép erőforrásait, és elosztja azokat a felhasználók között. Lehetővé teszi, hogy a felhasználók a programokat futtatásukhoz, vezérli a géphez kapcsolt perifériákat, fájlrendszerét van az adatok, a programok és a dokumentumok tárolására.

Tágabb értelemben nemcsak a magot nevezik UNIX-nak, de beleértik az olyan fontos programokat is, mint a fordítókat, szöveg szerkesztőket, parancsnyelveket, a fájlokat másoló és kinyomtató programokat. Sikeresen titka az, hogy C nyelven íródott, a mikroszámítógépektől kezdve bármely kiépítésű számítógépen futtatható, s másrészt a hozzáférhető és magas szintű nyelven íródott forráskód egyszerűen teszi a rendszernek még a különleges követelményekhez való illeszkedését is. Vegyük a UNIX jó operációs rendszer, különösen programozóknak.

A könyv a UNIX filozófiájával kívánja

megismertetni az olvasót. Mivel e filozófia a programok kapcsolatára épül, a különböző fejezetek elsősorban az önálló eszközöket vizsgálják, tárgyalják a programok összekapcsolásának kérdéseit és azt is, hogyan lehet a programokat újabb programok építésére használni.

A könyv nem referenciakönyv, ezért főleg a UNIX alkalmazási stílusára tanít. További információk megszerzéséhez a UNIX Programozói Kézikönyv szükséges.

**Honerkamp, M. — Jetter, M.:
 Repülés mikroszámítógéppel**
 (Spectrum, Commodore 64,
 Apple II, IBM PC)
 (Budapest, 1988.
Műszaki Könyvkiadó, 212 oldal.
Ára: 98,— Ft.)

„Ennél csak a valóságos repülés szebb!” — írhatják a szerzők könyvük előszavában, hiszen szinte minden olyan, repüléssel kapcsolatos tudnivalót sorra vesznek, melyeknek számítógépes szimulálásával a képernyő előtt ülők egy igazi pilótafülkében érezhetik magukat. A kötet a „Flight Simulator II”-vel való repülést ismerteti. A bemutatott repülési manőverek és szabvány repülési eljárások minden más repülés-szimulációs programra is alkalmazhatók.

A könyv célja, hogy betekintést nyújtson a repülésbe, és egyúttal élvezetesebbé tegye a repülés-szimulátorral való szórakozást. Minden fejezet a szimulációs program, illetve a repülés egy-egy részfejezetét tárgyalja. Ha az olvasó már gyakorlati repülő, néhány fejezet átlapozásával mindjárt a vakrepülésre térhet át. Az élvezetes repüléshez a szükséges repülőterképeket is megtalálja a kötet végén.

Lajstromolt robotok

A MTESZ-ben működő Magyar Robottechnikai Társaság a KSH segítségével megvizsgálta, hogy hazánkban hány robot dolgozik, s ezek milyen munkát végeznek. A gépiparban 26 vállalat vásárolt robotot. Ezek közül hét különféle gépeket, hat járműveket, öt villamos gépeket, három fémtömegcikket, illetve hirodatéchnikai eszközöket, négy műszereket, egy pedig vaskohászati termékeket gyárt. A robotokat csaknem húszféle munkára alkalmazzák: a legtöbbet ivégasztásra, különféle szerszámok kiszolgálására, présontásra, továbbá anyagok felületkezelésére. Használnak még robotokat öntvények tisztítására, anyagok lézeres megmunkálására, illetve mechanikai és elektronikai szerelésre.

A Magyar Robottechnikai Társaság az adatok alapján az érdeklődőket azonnal informálni tudja arról, hogy egy-egy speciális robot mely vállalatnál dolgozik. A tájékoztatást megkapta a társaság 35 tagvállalata s 350 egyéni tagja. A Magyar Robottechnikai Társaság vállalja, hogy ezentúl folyamatosan gyűjt információkat a működő ipari robotokról.

Fakultáció

Lentiben, a 2-es Számú Általános Iskolában 1985-ben számítástechnikai fakultáció indult, ehhez korszerű szaktantermet is kialakítottak. A célnak megfelelő, praktikus pultrendszeren tizenegy Commodore 16 és két Commodore Plus/4 típusú személyi számítógép várja a tanulókat. Az egyik gépet a tanár használja.

A központi gép összeköttetésben áll egy „elektronikus táblával” — azaz egy színes televízióval —, így a diákok nyomon követhetik a tanári modellprogramokat és magyarázatokat. A tanulók Junoszty típusú monitorjain láthatják programjukat, s ellenőrzik az esetleges hibákat.

A hetedik és a nyolcadik osztályosok közül 27 tanuló heti két-két órában sajátítja el a számítástechnika alapjait. A számítástechnikai oktatást vezetó Fracz József matematika — technika szakos tanár elősorban a diákok algoritmusos gondolkodását fejleszti. És a foglalkozásokon megismerkedhetnek a gyerekek a számítástechnika alapfogásaival és a személyi számítógép működési elveivel is.

A szaktanterem kialakításában a szülők áldozattalalásán kívül meghatározó szerepet játszott az EFPA, a Fém- és Faipari Szövetkezet, a költségvetési üzem, valamint a megyei és a városi tanács. Együtt több mint félmillió forintot adtak a vállalkozáshoz.

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közületeknek gépetl soronként (60 karakter) 100,— Ft., magánszemélyeknek az első sor 50,— Ft., minden további sor 20,— Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

Amiga, Plus/4, C16 szoftverek eladók. Rendelkezem minden programmal, ami Magyarországon létezik ezekre a gépekre. Eladók Atari ST Magazin (NSZK) eddig megjelent példányai. Érdeklődni: 848-471
Amiga mikrodíszek, a rajtuk levő 20 db új játékos és felhasználói programokkal eladók. Dr. Krón István, Debrecen, Dózsa Gy. u. 3. Tel.: 25-171

Amigások! A legújabb játékgéprokat olcsón megszerzhetitek. Egy lemez átvételre 50 Ft. Érdeklődni lehet: Budapest, Laky-köz 11. 1142, Tel.: 643-452

Atari 800X (új) magnóval eladó. C64 csere érdekel. Pintér Henrik, Mágocs, Gagarin u. 1/a. 7342

C16, C Plus/4, C64-es programok eladók. 10 Ft/db + utánvétel. Tel.: 06/429-168

C Plus/4 gépéhez 8 és 16 k-s epromok égetéséhez-olvasáshoz alkalmas programot, valamint építési leírás adok. Tersánszky Csaba, Miskolc, Engels u. 8. 3529, Tel.: (46) 65-133, 17-től 20 óráig
C64-re 1987/88-as programokat adok (20 Ft/db) és cserélkedni kezettán. ifj. Varga Péter; Nyíregyháza, Óvoda u. 49. 4400

C64 + 1541-II floppy + magnó + 2 db Joy + Junoszty monitor (TV-nek is használható) + 40 db kétdoldalas 3M-es lemez + sok szakkönyv + sztereosított interfész + GEOS 1.3-as verzió (német) + magyar nyelvű leírás kedvezményes áron eladó. Sürgős! Figyelem! A kb. 700 db felhasználói és játékos, valamint oktató prg-ot ingyen adom hozzá. Leveleket árajánlatokkal a következő címre kérem: Csík László, Újváros, Dózsa Gy. u. 23. 5052

C64-re programok eladók (kazettán). Zsámok Balázs, Nyírbátor, Édesanyák u. 28. 4300

Tárolható a kész kép

A tizenötézer dolgozót foglalkoztató leningrádi Lenelektronmas tudományos-termelői egyesülés a közelmúltban már IBM PC-vel kompatibilis személyi számítógépeket is gyárt. Az Iszkra 1030M típusú gépéhez videoképet feldolgozó programot fejlesztett ki. A videokamerával rögzített képet a számítógép segítségével átalakíthatjuk, retusálhatjuk, módosíthatjuk, sőt feliratozhatjuk is. A képet végső formájában mátrixnyomtatóval jeleníthetjük meg. Ugyanakkor lehetőség van a digitális formájú képre a mágneses adathordozón való tárolására is.

Az Iszkra 1030M-mel készített, feliratozott portré



TANFOLYAMOK — ENTERPRISE gépre

Kezdőknek

- 3 napos intenzív 1800.— Ft
- 4 x 4 órás alap (délutánonként) 1800.— Ft

Haladóknak

- Gépi kódú programozás — 32 órában 3200.— Ft

Jelentkezés, felvilágosítás:

Bakó Lászlónénál

NOVOSCHOLL

Bp. XIII., Kresz Géza u. 14.

Tel.: 122-099, 122-095

DOK — VESZEK — CSERÉLEK

C64 programokat eladók, lemezen és kazettán. 1988-as programokat cserélek. Micsik Gábor, Szeged, Szent Miklós u. 3. Tel.: (62) 27-771

C128, 1570-es floppy, magnó, 2 db joystick, 20 disk a legújabb programokkal, sok szakkönyvvel kedvező áron eladó. Antal Attila, Győr, Zöld u. 25. 9028. Tel.: (96) 24-735

Commodore 1280 (3 elsődleges üzemmód: C128, C64, CP/M; beépített 1571-es floppy) és 120 db disk a legújabb programokkal olcsón eladó. Tegyen ajánlatot! Radics Róbert, Kiskunfélegyháza, Szőlő u. 10/4. 6100

Commodore datasett adatmagnó (3000 Ft) és 32 kb-átos cartridge (2200 Ft) kifogástalan állapotban eladók. Muzsik József 587-835 este

Eladó: 31 BIT-LET, 55 Mikrovilág (programlistával), 40 Mikromagazin, 3 könyv + prospektusok. 1000 Ft + utánvétel. Tóth József, Hajdúböszörmény, Tyerskova u. 22. 4220

Enterprise programokat adok és cserélek. Széles programválaszték. Válaszboríték ellenében lista! Cím: Sándor József, Bonyhád, Bezerédi u. 41. 3/5. 7150

Enterprise (128 k) számítógép magnóval, programokkal, kézikönyvekkel 10 000 Ft-ért eladó. Szarvas Imre, Dunakeszi, Jászai M. u. 20. 2120

Fordításokat készíték a 64'er Magazinból. Több mint 3000 oldalnyi kész anyag! Kedvező árak. Textomat+, GEOS 2.0, Hi-Eddi, Giga CAD, Disc Wizard, Disc Demon stb. dokumentációk. Szolnoki Béla, Budaörs, Pf.: 400. 1446. Válaszborítékot kérünk!

Hardver, szoftver ötletek, programok cseréjének közvetítése. SYS-TEAM, Kecskemét, Pf.: 57. 6001

L.C.M. — THE NAME OF QUALITY

L.C.M. — a legújabb és leggyorsabb ellátó, programújdonságokkal:

— Commodore 64 (disk/kazetta): kb. 50 program havonta

— Amiga: több mint 30 disk a legújabb programokkal havonta

— Atari ST: több mint 20 disk és több mint 30 program havonta

L.C.M. biztosítja az Ön ellátását a legújabb programokkal a felsorolt géptípu-

sokhoz. Levelezek angolul vagy magyarul: Milosevic Slobodan, 19000 Zajecar, Naseslje Avnoj C1/1-39, Jugoszlávia, Tel.: 019-21010

Primo SSD-hez 32 k-s komplett felhasználói programcsomag égetése. Vidékre utánvétellel. Előzetes megbeszélés: Horváth László 583-544 délelőtt

PRIMO-sok figyelme! Eladók Primóhoz hardverek és programkiszételek! Több mint 1000 program! Kívánságra részletes listát küldök, válaszborítékot kérek! Varsányi Gábor, Nagyatád, Aradi u. 1/IX/C. 7500

Seikosha GP 700/A 8 színű grafikus mátrixprinter olcsón eladó. Érdeklődni lehet nappal a 418-166, este a 312-183-as telefonon Kovács Lászlónál.

SPECTRUM programok (48 k) eladók. Magyarországon a legolcsóbban, 12 Ft/db. Válaszborítékért katalógust küldök. Megfelelő partner esetén cserével is foglalkozom. Eötvös Levente, Debrecen, Cserepes u. 22. 4026

Spectrum+ (80 k), interfész II., magnó, 15 kazetta, 10 könyv eladó. Új 3,5" lemezek (Real data) eladók. Érdeklődni: Polgár József, Kecskemét, Balaton u. 20. fsz. 30. 6000. Tel.: (76) 47-269

ZX-Spectruma programok eladók! Minden a legújabb szoftverek! Ingyenes katalógus! Pilláry Gábor, Pécs, Bajcsy-Zs. u. 4. 3/9. 7622

ZX-Spectrum+ 48 k-s számítógép, Kempston interfésszel, szakkönyvekkel, 300 db felhasználói és 600 db játékprogrammal együtt 15 000 Ft-ért eladó. Kápolnási János, Veszprém, Felszabadulás u. 63/A. 6/38. 8200

Z80 alapú CP/M op.r. mikrogépet dual 5'25" floppyval (800 kb-át/disk) Philips monitorral (igény esetén eprom programozó adapterrel) és CP/M szoftverekkel olcsón eladó. Az érdeklődéseket nappal a 418-166, este a 312-183-as telefonon Kovács László várja.

Tudja Ön, hogy milyen egyszerű a FASTLOAD CARTRIDGE-el dolgozni? Vállalom bővített karakterkészletet, Reset gomb, IRO (pause) kapcsoló beépítését, Seikosha SP1200VC - 180 VC nyomtatók magyarosítását, FASTLOAD, SPEEDTAPE programsegítő modulok készítését. Szívesen

küldök tájékoztatót! Bártfai Barnabás, Budapest, Möricz Zs. u. 22. 1193. Tel.: 668-411/64 (délelőtt).

VENZEK

Sinclair ZX81 számítógépemhez billentyűzetet keresek. Siófok, Somlói u. 1. 8600. Tel.: (84) 12-431

CSERÉLEK

C16, C Plus/A, C64 programokat cserélek. Listát kérek. Lékó Zoltán, Szolnok, Dr. Csánádi krt. 8. VI/18. 5000

Commodore 16-os, 64-es, 128-as géppel rendelkezőkkel kapcsolatot keresek, hét-ezer programom van. Amigához 700 lemeznyi programom van. Gyermán Sándor, 23000 Zrenjanin Rade Kóncra 23/V. Jugoszlávia

C Plus/A felhasználói programokat cserélek, kizárólag lemezen. Kálmán Albert, Eger, Rákóczi u. 31. III. 11. 3300

C64-es programokat cserélek kazettán. Budapest, Országbíró u. 4. IX. 38. 1139

C64-es programokat cserélek lemezen. Minőségi választék, leírások is érdekelnek. Válaszokat listával kérem, én is küldök. Péli Gábor, Budapest, Tomori u. 10. 1131. Tel.: 496-032

C64 szuper programokat cserélek kazettán. Listát kérek! Purnhauser Pál, Tárkon, Templom u. 63. 2461

C128 programokért C64 programokat adok. Ugyanitt C64 programok cseréje, lemezen. Kis Bogdán Zsolt, Nagypeterd, Kossuth u. 126. 7912

Enterprise programokat cserélek kazettán. Sántha József, Budapest, Vadaskerti u. 8. fsz. 2. 1021

MSX rendszerű számítógéphez programokat cserélek, néhányat eladok. Jancsicsk István, Miskolc, Dráva u. 7. 3528. Tel.: (46) 12-634

Primo programcsere! Több, mint 1000 program! Csak levélben! Listát kérek! Vajai János, Budapest, Nagytétényi u. 234. J. 2/8. 1225

ZX-Spectrum programokat cserélek. Listát kérek! Székely Gyula, Debrecen, Apáfa 14. Pf.: 2649. 4002

Ez a rovatunk **KODEX 2000** szövegszerkesztővel készült.

Az Olvasó írja

Úgy látszik, hogy folyamatos invitálásra egyre több olyan olvasói levelet kapunk, ami nemcsak dicséret vagy elmarasztal, de fontos információkat is közöl. Nem titkolt vágyunk, hogy minél több olvasó bevonjunk az újságírásba, hiszen ha sokan írják a lapot, akkor valószínűleg gazdagabb lesz, mint ha csak a szerkesztők írásait közöljük. Lássuk a leveleket.

Vass Tamás

Lapjuk rendszeres olvasója vagyok. Főleg az Adok – veszek – cserélek rovatot böngészem. Most mintegy 130 programom van, ebből csaknem 50 programot cserélték ki. Akiknek eddig írtam, azok közül kettőn választott tizenötöt. Ha a szerkesztőséget érdeklí, szívesen rendelkezésükre bocsátok POKE-okat, valamint programokat. Az októberi számban örömmel olvastam azt a Bioritmus nevű programot, ami C Plus/4-re készült. Be is írtam, de a program hibás volt! Ha a születési dátumot 1 nappal megváltoztatjuk és utána megjelenítjük az eredményt, akkor az a csik, amely a mai napot jelzi, 5 nappal ugrik odébb, nem eggyel. Hasonló hiba volt az augusztusi számban közölt Analóg órával. Ezt a programot javítottam, s ha érdeklí öntek, elküldöm. A számlapra négyféle variációt készítettem. Kérem, írjon olyan címekeket nekem, amelyekre ha írok, cserélhetek C16, illetve C Plus/4-es programokat vagy vehetek.

Sajnos néha előfordul, hogy hibás programot adunk közre, ezért szívesen fogadjuk és közöljük is olvasóink észrevételeit. Ami a kérését illeti, nem gyűjtjük a címekeket, de szívesen közöljük a cserére szülő felhívását, ám ehhez ismernünk kellene a postai címét, hogy azt az újságban közreadjuk.

Váraljai Miklós, Budapest

Immár hat éve vagyok olvasója a lapnak, és szinte csak jót tudok mondani róla. Míg az első számok cikkei még inkább csak a figyelemfelkeltésre irányultak, az újság mostani felépítése, hűgadsága egészen elsőrendű. Nagyon jónak találom például, hogy a játékprogramok rovat megszünt (?), ugyanis az ilyen, esetleg féldolgalas játékprogramok színvonala csakis sokkal alacsonyabb lehet, mint azoké a programoké, amiket bárki megszerezhet. Az a játékprogram, amit valaki több órai rabszolgamunkával bepötyög (nem beszélve a hibákról) és utána hamar, néhány órán belül megun, nem sokat fejleszt az illető programozási képességeit. Talán ezután majd gépnél tud jobban. Viszont nagyszerűnek tartom, hogy olyan programok használatát ismertetik, amelyek általában elterjedtek, és használatukra igény is lenne. (A kalózmásolás hátránya, hogy nincs leírás.) Baj, hogy csak nagyon nehezen lehet beszerezni az ilyen leírásokat. És ráadásul még olyanok is vannak, akik ezekért horribilis összegeket kérnek, amit néha a felhasználó jobb híján meg is fizet. Az ilyen kalózmásolatról csak annyit, hogy nem mindig a másolatok kellene szidni. Úgy öt éve az egyik termelőszövetkezet az újonnan vásárolt C64-hez vett egy programot (állami

nagyáruházban), mintegy 60 ezer forintért. A programot volt szerencsém átnevezni: teljes egészében megegyezik a Simon's BASIC-kel. A különbség csupán annyi, hogy az utasítások magyar nyelvek és ékezetes a karakterkészlet. Az ilyen programokat miért nem ellenőrzik? Különben pedig a jelenlegi árak mellett nem csoda, ha inkább másolnak a számítógép-tulajdonosok.

E folyamat ellen szerintem ma már nem sokat tehetnek. Úgy vélem, ma nincs olyan számítógép-tulajdonos, akinek ne lenne ilyen másolt programja. Visszatérve az újságra, nagyon örömdetes, hogy elkezdtek a számítógép egyéb felhasználási lehetőségeivel is alaposabban foglalkozni, gondolok itt a mérésre és vezérlésre vagy az irányításra. Bizonyára ez olyan téma, amelyről nem írhatnak elégyszer, és ennél a témakörnél nem szabad sajnálni a magyarázatokat, mert ha valaki valamit a gyakorlatban ront el, annak akár a gép „életébe” kerülő következményei is lehetnek.

Végül lenne egy kérésem. Augusztus óta élvezem a Magyar Néphadsereg vendégszeretetét, és abban az első hónap nagy kapkodásban lemaradtam a szeptemberi számról. Nagyon szeretném, ha megmondaná, hol lehet beszerezni ezt, mivel eddig az összes lapszám megvan.

A leirtakhoz nincs sok hozzáfűzivalom, a véleményünk egyezik. Ami a régebbi példányok vásárlását illeti, forduljon bizalommal az NJSZT titkárságához (Bp. V., Báthori u. 16.).

Baráz Tamás, Maglód

Mint tudják, a Sinclair gépyárnak sok terméke van a ZX81-en és a Spectrumokon kívül, ha azok az enyhén szólva hiányos magyarországi dokumentáció és főlajtás híján nem is lettek olyan népszerűek, mint az előbbieik (C64, C16 stb.). A Sinclair QL szinte az összes gépet lepiálja magyar–angol karakterkészletével, Super BASIC programozási nyelvvel, melyben beépítve megtalálható a fől, le, jobbra, balra scroll rutin (1 parancs!), belső óra, melyet egy paranccsal így ír ki például: 1988 JAN 1. 12:21:30, maradékszámoló függvény, beépített fill (kitöltés), beépített teknoB&G-grafika, számítógép-hálózatra csatlakozás, külön vezérelt ablakok, két beépített lemezmaghajtó, 512 x 256-os felbontás, 256 szín, külön QL monitor, átszínezés, körök, ellipszisek, görbék, vonalak relatív és „sima” koordináta-rendszerben, kilvintantásos, villámgyors grafika, és hangja is van, de még milyen!

12-féle csatlakozási lehetőség, interfész nélkül! Fájlképezés, 111-120 különféle utasítás, újrastartozás, kilépés gépi kódban, kényszerelv, ROM cartridge csatlakozó (ez az egy nem BASIC). Mindez BASIC!

Ezenkívül a gép kapható 128 k, több Mbájtos kisereléssel winchesterekkel, floppykkal, ingyen szövegszerkesztő diagram, archívum, költségvetés-táblázatok. Mindezt annyitért, mint a C64. Hát nem csodálatos? És ezt nem ismerik! Erről nem ír senki? Írjanak róla valamit! Vagy

ezt a levelet közöljék, ha mód van rá! Sok programot tudok adni, ami közölni lehet, csak előbb szerzek egy nyomtatást.

Igazán élveztem a Sinclair QL-ről írt sorait, de erről nekem az a véleményem, hogy „kinek a pap, kinek a papné”. De feltréve a humorizálást, nem tudom, mennyi QL van az országban, mindenesetre hozzánk gyakorlatilag nem érkezik QL program, amit közölnék. Ha tud programot adni, és ezek érdekesek, saját maga által készített programok és ábrák is, akkor nem hiszem, hogy ne közölnek belőlük néhányat.

Patek Alajos, Budapest

Nemrégiben levélben ismerttettem önökkel a Novotrade FINE PEN című kezeltájával kapcsolatos rossz tapasztalataimról, és valamilyen segítségüket kértem. Aztán telefonon felhívott a Novotrade-tól Molnár Attila, és — elismerve, hogy a program kezelési utasítása nem kielégítő — ígért tette, hogy egy héten belül megküldik részemre a részletes használati utasítást. Úgy gondolom, kötelességem ezt önökkel is közölni, miután bizom abban, hogy az úgy ezzel lezárul.

Köszönjük az értesítést.

Varga Péter, Budapest

Nem szokásom újságoknak levelet írni, de ha már ügyis küldöm a megrendelőt, akkor fűzők hozzá a kis dicséretet és megírást is. Az első szám óta rendszeres olvasója vagyok az újságnak, talán nem is volt olyan lapszám, amit elmulasztottam volna. Majdnem mindegyikben találtam valamit, ami számomra is érdekes volt, bár bevallom, hogy engem egyáltalán nem érdekelnek a játékprogramok és a csak mikrogépeken hasznosítható dolgok, lévén, hogy kezdetől fogva IBM PC gépen volt módom dolgozni. Viszont nagyon tetszik, hogy önök sok olyan cikket közölnek, amelyek nem egy géptípusra való programok, hanem eljárások, algoritmusok, elvek és általánosan hasznosítható ötletek leírásai. Külön örülök annak, hogy az idő haladtával ezeknek a cikkeknek az arányát nőni látom. Szeretném, ha több IBM PC-vel kapcsolatos cikk jelenne meg (igérem, ha tudok, küldök valamit), és minél több általánosan hasznosítható írás. A könyvismertető cikkek általában nagyon sokára jelennek meg (néhányik könyv már el is fogy addigra), és akkor is elég szűkszavúak. Talán lehetne szólni a könyvek minőségéről is: hasznos-e, érthető-e, érdemes-e megvenni vagy nem ér semmit.

Mindig öröm, ha az olvasók észreveszik és nyugtázzák a szerkesztők törekvéseit. Valóban az a helyzet, hogy erősen mérsékeljük a játékprogramok közlését, mert úgy látuk, a legtöbb olvasó jobban szeret figyelemfelkeltő cikkeket olvasni, mint szolgálaton bepötyögni néhány tíz vagy száz utasítást. Lassan elkezdett a lap a PC kompatibilis gépekkel foglalkozni, és eljön az az idő, amikor egyes korszerűtlen gépekkel már nem is törődünk mi sem.

Valamennyi olvasónknak szívélyes üdvözlémet küldöm:

Kovács Győző



A M. A. D. rendszerrel készített polaroid foto

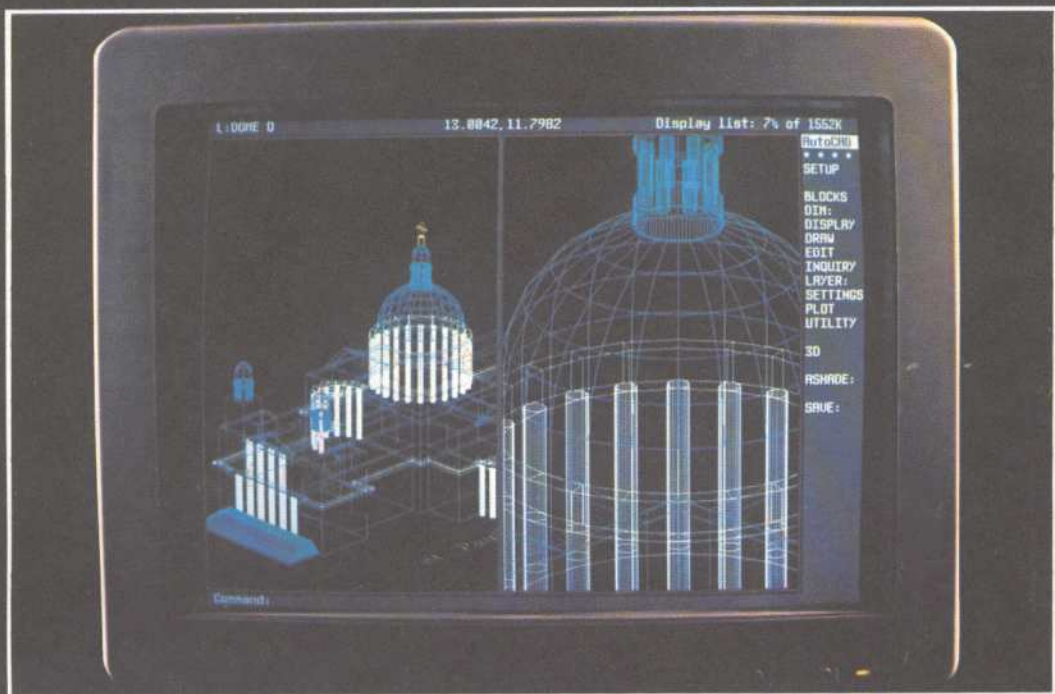
Digitalizált retus

Nemsokára vége a kézi retusálásnak — állítják a fotópiac szakértői —, legalábbis ami a fekete-fehér képeket illeti. Amerikában olyan „számítógépes fotólaborokat” dobtak piacra, amelyek segítségével néhány gombnyomással eltüntethető a fényképről a nagymama fejéből kinövő karácsonyfa vagy a tájképben éktelenkedő villanyoszlop. A dolog nyitja: a képeket digitálisan ábrázolják, így a jeleket sokkal könnyebb megváltoztatni, mint a fotópapíron vagy a filmen, ahol kémiai beavatkozásra van szükség. De hogy kerül a fénykép a képernyőre? Ennek két útja is van. A gyakoribb az, hogy a

már kész papírképet bedugják egy 2500 dolláros képolvatóba, amely a fotót úgy pásztázza végig, mint egy fénymásológép. A képernyőre azonban kerülhet olyan kép is, amely soha nem látott fotópapírt; néhány éve ugyanis léteznek olyan fényképezőgépek, amelyek a képet nem filmre, hanem hajlékony mágneslemezre rögzítik. Egy lemezen ötven kép tárolható. A lényeg tehát az, hogy a fényjeleket digitális jelekké alakítják át. A képernyőn látható képet ezután kedvünkre alakíthatjuk, s a végleges változatot ismét papírra rögzíthetjük, feltéve, hogy van egy legalább 3 és fél ezer dollárba ke-

rülő, jó minőségű nyomtatónk. A gépi retus egyelőre nem tartozik az olcsó mulatságok közé, de az amerikai fotósok körében már népszerű az új felszerelés. Az árak azonban gyorsan csökkennek. Egy mágneslemezrel működő fényképezőgép ára például jelenleg 2700 dollár, de a gyártók azt ígérik, hogy egy év múlva már 1000 dollárért is lehet kapni.

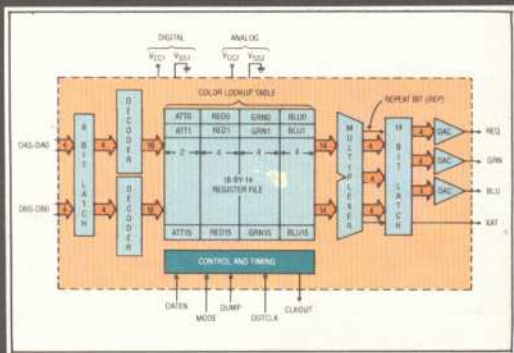
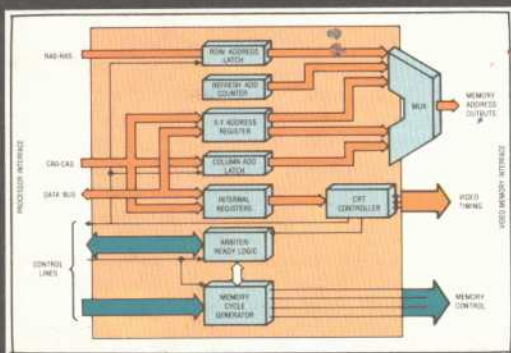
Érdekeség, hogy Párizsban a M. A. D. cég már szolgáltatásként kínálja az Amiga vagy PC képernyőről — diapozitívrá, illetve polaroid papírra — digitális úton készített színes fotót. Az utóbbi egy perc alatt készen van . . .



A Verticom HX kártyával és TwinFocus szoftverrel készített kép

1. ábra

2. ábra



A TMS34061
videorendszer-vezérlő folyamatábrája

A TMS34070
színes paletta folyamatábrája